Prime 2.1

User Manual

Schrödinger Press

June 2009

Contents

Document Conventions

In addition to the use of italics for names of documents, the font conventions that are used in this document are summarized in the table below.

Font	Example	Use
Sans serif	Project Table	Names of GUI features, such as panels, menus, menu items, buttons, and labels
Monospace	`$SCHRODINGER/maestro`	File names, directory names, commands, environment variables, and screen output
Italic	*filename*	Text that the user must replace with a value
Sans serif uppercase	CTRL+H	Keyboard keys

Links to other locations in the current document or to other PDF documents are colored like this: Document Conventions.

In descriptions of command syntax, the following UNIX conventions are used: braces `{ }` enclose a choice of required items, square brackets `[ ]` enclose optional items, and the bar symbol | separates items in a list from which one item must be chosen. Lines of command syntax that wrap should be interpreted as a single command.

File name, path, and environment variable syntax is generally given with the UNIX conventions. To obtain the Windows conventions, replace the forward slash / with the backslash \ in path or directory names, and replace the $ at the beginning of an environment variable with a % at each end. For example, `$SCHRODINGER/maestro` becomes `%SCHRODINGER%\maestro`.

In this document, to *type* text means to type the required text in the specified location, and to *enter* text means to type the required text, then press the ENTER key.

References to literature sources are given in square brackets, like this: [10].

Introduction

Prime 2.1 is a highly accurate protein structure prediction suite of programs that integrates Comparative Modeling and Fold Recognition into a single user-friendly, wizard-like interface. The Comparative Modeling path incorporates the complete protein structure prediction process from template identification, to alignment, to model building. Refinement can then be done from a separate panel, and involves side-chain prediction, loop prediction, and minimization.

The Prime interface was designed to accommodate both novice and expert users, while the underlying programs were designed to produce superior results in a variety of applications, including:

- High-resolution homology modeling
- Refinement of active sites
- Induced-fit optimization
- Fold Recognition
- Structure-based functional annotation
- Generation of alternate loop conformations

1.1 The Prime Suite in Maestro

To run Prime you use Maestro, the graphical user interface (GUI) for all of Schrödinger's products. For an introduction to Maestro, see the *Maestro Overview*. For help with a Maestro panel, click the Help button. For more information, see the *Maestro User Manual*.

The Prime submenu of the Maestro Applications menu has four items, Structure Prediction, Refinement, Covalent Docking, and MM-GBSA, which provide access to the Structure Prediction (Prime–SP), Refinement, covalent docking and MM-GBSA modules of the Prime suite.

Prime–SP consists of a series of steps leading from a protein sequence through Comparative Modeling to the construction of a 3D structure. If a template of sufficiently high sequence identity exists in the database of known structures, the Comparative Modeling path is used. If a suitable template cannot be identified using sequence information, however, Fold Recognition can be used to identify potential templates.

Prime–Refinement is a facility for refining protein structures that have been imported into Maestro or added to the Project Table from the Comparative Modeling workflow or another Schrödinger application.

Covalent Docking is a facility for “docking” ligands that are covalently bound to the receptor. It uses designated reactive bonds in the ligand and a receptor bond to sample the possible attachment points, and performs a loop prediction to sample the ligand and specified parts of the receptor. The result is a set of “poses” of the ligand, docked to the receptor.

Prime MM-GBSA is a facility for calculating ligand binding energies within the MM-GBSA continuum solvation model.

1.2 Prime Documentation

The *Prime User Manual* describes the Prime suite, including information about:

- Essentials of using the Prime–SP module: runs and projects, job control, and the layout and common features of step panels
- Starting a structure prediction: the Input Sequence and Find Homologs steps
- Using the Comparative Modeling path, including the Edit Alignment and Build Structure steps
- Using Fold Recognition
- Using the Prime–Refinement module
- Running Prime MM-GBSA calculations
- Running Prime jobs from the command line

This document expands on the information provided in the Maestro online help. In addition to describing steps, features, and options, this document suggests ways to determine which choices are the most useful, describes results, and provides some background and technical detail. Terms pertinent to Prime are defined in the Glossary.

For a tutorial introduction to Prime, see the *Prime Quick Start Guide*. For information on installing Prime, see the *Installation Guide*.

For information about Schrödinger’s Induced Fit Docking protocol, which uses Prime and Glide to optimize docking by including ligand-induced flexibility in the receptor active site, see the document *Induced Fit Docking*.

Maestro online help includes topics for Prime. See page 131 for a discussion of help in Maestro, including tooltips (Balloon Help) and technical support contacts. For a fuller description of the help facility, see Chapter 15 of the *Maestro User Manual*.

1.3 Running Schrödinger Software

To run any Schrödinger program on a UNIX platform, or start a Schrödinger job on a remote host from a UNIX platform, you must first set the `SCHRODINGER` environment variable to the installation directory for your Schrödinger software. To set this variable, enter the following command at a shell prompt:

csh/tcsh: `setenv SCHRODINGER` *installation-directory*

bash/ksh: `export SCHRODINGER=`*installation-directory*

Once you have set the `SCHRODINGER` environment variable, you can start Maestro with the following command:

```
$SCHRODINGER/maestro &
```

It is usually a good idea to change to the desired working directory before starting Maestro. This directory then becomes Maestro's working directory. For more information on starting Maestro, including starting Maestro on a Windows platform, see Section 2.1 of the *Maestro User Manual*. There is no need to set `SCHRODINGER` on Windows: it is set automatically.

Prime jobs, with the exception of Fold Recognition, can be launched from a Windows platform to a remote UNIX host. You can run Prime MM-GBSA jobs locally on a Windows host.

To launch Structure Prediction jobs from Maestro, you must have access to the PDB. If the PDB database is not installed into `$SCHRODINGER`, you must set the `SCHRODINGER_PDB` environment variable to the path to the PDB installation. For information on how to set environment variables on Windows, see Appendix A of the *Installation Guide*.

1.4 Citing Prime in Publications

The use of this product should be acknowledged in publications as:

Prime, version 2.1, Schrödinger, LLC, New York, NY, 2009.

Using Prime–Structure Prediction

This chapter presents the essentials of using Prime–Structure Prediction (Prime–SP):

- Starting, naming, and saving Prime–SP runs in Maestro projects
- Navigating within and between runs
- Job control, job monitoring, and Prime–SP job options
- Using Prime–SP step panels, including their common layout and features
- Using the Prime menu bar and Prime toolbar

2.1 Prime Runs and Maestro Projects

A single execution of the Prime workflow using a particular set of choices of templates, paths, and settings is called a *run*.

Prime runs are stored within Maestro projects. Structures generated in completed Prime runs are added to the Maestro Project Table. If you are unfamiliar with Maestro projects and the Maestro Project Table, you may want to review the Project Table and Project Facility online help topics. For more information, see the *Maestro User Manual*.

Maestro always has a project open. If you begin to work without selecting or naming a project, Maestro creates a *scratch* (unnamed) project.

Prime always has a run open, which belongs to the current project (whether it is a named project or a scratch project). If you begin the Prime workflow without specifying an existing run, the default name for the current run is `run1`. Multiple runs can be performed within a project, but you must save the project in order to save the data in the runs.

Each project can be displayed in the Project Table panel. In Prime, Project Table entries are usually finished model structures with their properties. Prime stores data in the project as each step in a run is performed, but most of this data is not displayed in the Project Table, which can remain empty until a run is completed. For this reason you should save scratch projects in which you have done work, even if the Project Table is empty.

Two Maestro projects can be merged into one. If you merge two Maestro projects, each of which contains one or more Prime runs, all the runs are included in the merged project. If two runs have the same name, Maestro automatically makes the names unique by appending `-mrg1` to one of them.

Maestro projects containing multiple runs are useful for comparing the results of workflows in which different selections and options were chosen: for example, a different template selected for building, or a job performed with a different set of options. Creating these multiple runs may involve navigating to an earlier step in your current run, making different selections, and then navigating forward again in a new run. For a description of the features used in navigation (the Next button, the Back button, and the Guide), see Section 2.2 on page 7.

If you return to an earlier step in your current run and make different choices as you proceed forward, you will be asked whether you want to overwrite your existing data. If you want to keep the data in your current run for comparison rather than overwriting it, you can start a new run. The commands for saving the old run, starting a new one, and switching between Prime runs are found under the Prime File menu.

The File menu commands for managing runs are:

New

Asks for a new run name (all run names must be unique), and starts a new run at the Input Sequence step.

Save As

Copies the current run under a new run name and switches to the new run. The original run is saved under the original name. If you open a new project and begin a workflow without explicitly giving the run a name, it will be given the default name of `run1`.

Rename

Asks for a new name for the current run. All run names must be unique.

Delete

Deletes the current run, on confirmation.

Open

Displays a list of options for switching to another run. The list includes the four most recently used runs. If there are more than four runs in the current project, click the More option to select a run by name.

To backtrack and try a different set of choices without overwriting your current workflow:

1. Choose Save As from the Prime File menu.

   This saves the current run under the old name (`run1` is the default for the first run in a new project) and creates a copy under the new name.

You are now in a new workflow identical to the first except for its name.

2. Navigate to the step where you want to make a different choice. If you want to go back more than one step, click the step name in the Guide.

2.2 General Panel Layout

Each step in Prime–SP has its own panel. The panels have many elements in common and share a standard layout. This section discusses these common elements, their location, and their function. An example of a Prime–SP panel is shown in Figure 2.1.

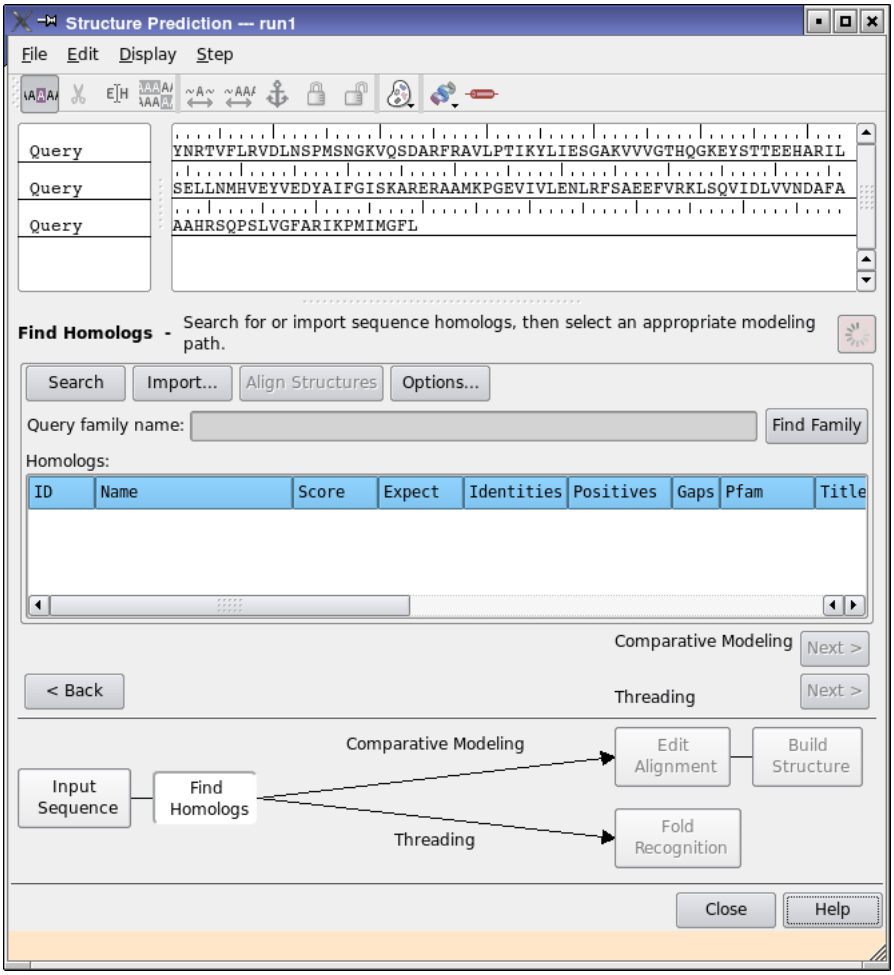

Figure 2.1. The Find Homologs step of the Structure Prediction panel.

In the upper part of the panel is the Prime menu bar, and below that is the Prime toolbar. Below the Prime toolbar is the Prime sequence viewer. You can resize the sequence viewer by dragging the small box in the right-hand corner of the horizontal line that separates the sequence viewer from the main panel. You can select a single residue in a sequence by clicking on it, and you can select multiple residues using shift-click to select a range and control-click to select or deselect individual residues.

In addition to displaying sequences for the query and for templates, in some steps the sequence viewer shows secondary structure predictions (SSPs). Right-click on an SSP for a menu of commands that includes Hide All, Delete, and Export. Right-clicking on a sequence also produces a shortcut menu of commands appropriate to the current step. You can export sequences using the Export Sequences command on the Prime File menu.

The sequence viewer includes a "ruler", which marks the residue position relative to the left-most residue displayed in the sequence viewer. This residue position is displayed in parentheses in the tooltip for each residue. The positions are arbitrary, and can change when you display residues in the viewer. They are intended to help you keep track of the sequence as you scroll the sequence viewer.

The sequence viewer can be configured to display a single line for the sequence that can be scrolled horizontally, or the sequence can be wrapped so that it scrolls vertically. The configuration is controlled by the Wrap Sequences option on the sequence viewer shortcut menu.

If you want to save an image of the sequence viewer, choose Save Image from the sequence viewer shortcut menu. A file selector opens, in which you can select a format, from PNG, TIFF, and JPEG, and save the image.

Below the sequence viewer are the step name and a brief description of the step, followed by a row of buttons. The first button on the left runs the main action of the step, which may involve launching a job. See Section 2.5 on page 14 for more information on running Prime jobs. An Options button is usually located to the right of the action buttons. Most buttons, including those in the Prime toolbar, the Prime Guide, and the Maestro toolbar, have tooltips that you can view by pausing the cursor over the button.

The job status icon in the upper right corner of the panel turns green and spins when you launch a job. When the icon turns pink again, the job is finished. Click the icon to open the Monitor panel. For more information on job control, see Section 2.5 on page 14.

The center of the panel contains the Step View, which shows data relevant to the step in table format. In the Build Structure step, the Step View contains output from the Build Structure job's log file. You can view the complete text of a truncated table entry as a tooltip by pausing the cursor over the text. To resize a table column, drag a column border using the middle mouse button. You can sort the tables in the Step View by clicking the column heading, once

for ascending order, twice for descending order. You can also export data from any table in CSV or HTML format from the table shortcut menu.

Below the Step View are the Back and Next buttons, the primary way to navigate between steps. You can perform a simple workflow by clicking Next after completing each step. However, it is sometimes useful to go back to previous steps, make different choices, and proceed forward again using the Next button or the Guide, explained next.

The Guide, below the Back and Next buttons, is a diagram of the Prime workflow as a series of steps, each step represented by a button. The diagram branches to show the steps of both paths, Comparative Modeling and Threading.

To hide the Guide, deselect the Guide check box in the Step menu.

The Guide has two purposes: to show your current location in the Prime–SP process, and to let you navigate back and review or change the input at previous steps in the process. The current step is highlighted in the Guide diagram. All previously accessed steps, forward or backward, are available. You can go to any of these steps by clicking them in the Guide.

Clicking on an available step button in the Guide displays the panel for that step. If a step is not yet available, its button is dimmed. Available steps include those that have already passed through in the current run, the present step, and (once step-specific criteria have been met) the next step beyond the current location. When the next step is available, clicking on it behaves the same as clicking the Next button.

Note: You cannot launch searches and jobs by clicking the steps in the Guide. To launch actions pertaining to a step, click the appropriate buttons within the step panel.

The Close button appears below the Guide on the lower left of the panel. To the right is the Help button, which opens online help related to the current step.

2.3 The Prime Menu Bar

The Prime menu bar consists of four menus: File, Edit, Display, and Step. Some of the options in these menus are also available as toolbar buttons.

2.3.1 The File Menu

The commands on the File menu for working with Prime runs (New, Save As, Rename, Delete, and Open) are discussed in Section 2.1 on page 5. The menu also contains the commands Export Sequences and Job Options. See Section 2.5 on page 14 for more information on job options.

2.3.2 The Edit Menu

The Edit menu contains commands for editing sequences and secondary structure predictions. These editing tools are also available from the Prime toolbar, and are described with the toolbar in Section 2.4 on page 13.

2.3.3 The Display Menu

The Display menu includes the following commands for controlling the appearance of the sequence viewer and structures in the Maestro Workspace.

Color Scheme

This command opens a menu of coloring options. In Prime–SP, the coloring schemes act as modes; that is, only one color scheme can be used at any given time. Each step in Prime has a default color scheme. You can use this menu to choose a different one. When amino acid sequences are added to the sequence viewer, they are colored according to the scheme that is currently selected.

When the structure associated with a sequence is displayed in the Workspace, it is generally colored using the same scheme as the sequence. However, protein structures in the Workspace are sometimes colored using other color schemes:

- When a PDB structure is imported, it is displayed in the Workspace using an error reporting color scheme in which complete recognized residues are gray, incomplete residues are red, unknown or nonstandard residues are orange, residues with multiple location indicators are green, and recognized residues with some unrecognized atoms are blue. For more information about conversion of PDB structures, see the *Maestro User Manual*.
- When the Build Structure step is complete, the color scheme applied to the structure in the Workspace is Atom PDB B Factor (Temperature Factor), in which blue atoms are those derived directly from the template, and red atoms have been predicted or modeled. To restore this color scheme, choose Atom PDB B Factor (Temperature Factor) from the Color all atoms by scheme Maestro toolbar button.

The Prime color schemes are:

- None: All sequences are colored standard gray.
- Color by Sequence: The query sequence is colored standard gray. Each of the other sequences uses a different color.

- Residue Type: Each amino acid sequence is colored according to the array of residue type colors.
- Residue Property: Each amino acid sequence is colored according to the array of residue property colors (fewer colors than residue type).
- Residue Matching: The query and templates are colored in a pairwise manner. Matching residues are colored according to their residue type, and nonmatching residues are colored standard gray. If more than one template is chosen, the query will have more colored residues because the effect is cumulative. If only the query sequence is shown, it is colored standard gray for all residues.
- Residue Homology: Aligned query and template residues that are identical (highly conserved) are colored red. Aligned query and template residues that are similar according to the BLOSUM62 scoring matrix (conserved) are colored yellow. Gray is used for residues with no homology.
- Secondary Structure: The query sequence is colored standard gray. All other AA sequences should have SSAs, which are colored according to the SSA colors.
- Template ID: Only available for the Comparative Modeling path. The query sequence is colored standard gray. The predicted structure is colored according to template ID for each region (using the cycle colors). Each template sequence is colored with its color, but only in the region that was used (the rest of the template is colored dark gray to indicate it is not used).
- Proximity: At any step in which the sequence viewer is displaying sequences that have associated structure (all steps except Input Sequence), color by proximity mode is available. When you select this color scheme, the current sequence colors do not change immediately. Instead, when the mouse is over a sequence that has structure, the cursor changes to indicate that proximity can be calculated. You can then select one or more contiguous residues in the sequence, and the system will color that sequence based on the proximity to the selected residues. No other sequences will have modified colors (even if they have structure too and are in proximity of the selected sequence).

  Proximity is calculated based on the shortest distance between any two atoms in two residues, not including the bonded C and N atoms in the residues next to the selected residues.

  To change the proximity cutoff, choose Preferences from the Display menu and change the number in the Proximity cutoff text box.

Font Size

The font size of letters and numbers in the sequence viewer can be changed using this option. The font size options are:

- Small
- Medium
- Large
- Huge

View Structure

The options for showing structures in the Maestro Workspace include:

- All
- None
- Predicted Only

View SSA

This option controls whether secondary structure assignments are shown in the sequence viewer. The default is not to show secondary structure assignments.

Legend

Choose this menu option to open the Legend panel, which includes the assignment of each color and symbol used in each color scheme and sequence representation. Click the Colors tab and select a Color scheme from the list, or click the Symbols tab and select a type of Sequence data, such as Pfam/HMMER or SSP. This option is also available when you right-click on a sequence in the sequence viewer.

Preferences

Choose this Display menu option to open the Proximity dialog box:

- Proximity cutoff: The value entered in this text box defines proximity for the Proximity color scheme. The default is 4.00 Å.

2.3.4 The Step Menu

The Step menu provides another way to navigate between steps in Prime. It includes the following options:

Guide

When this check box is selected, the Prime Guide is displayed (the default). Deselect the check box to hide the Guide.

Input Sequence

Click this option to go to the Input Sequence step.

Find Homologs

Click this option to go to the Find Homologs step.

There is an option to click to go to each step in Prime–SP. A red diamond is displayed next to the option to go to the current step. Options to go to steps that are not available are dimmed.

2.4 The Prime Toolbar

The Prime toolbar is the row of buttons just under the Prime menu bar. The Prime toolbar provides convenient access to commonly used Prime tasks. Pause the mouse over a button to see its description. These task modes are also available as options in the Edit and Display menus. The toolbar buttons are shown below as they appear on the toolbar, with their names and functions.

Select (Edit menu): Select a region of the sequence to see the corresponding residues highlighted in the 3D Workspace.

Crop (Edit menu): The cropping tool, which can be used in the Build Structure step to select ends of the query sequence that are not to be used for building.

Edit SSP (Edit menu): Edit secondary structure predictions. Highlight a region of the SSP, and then type either `H` for α-helix, `E` for β-strand, or `-` for loop.

Set Template Regions (Edit menu): If more than one template was selected in the Find Homologs step, use this button in the Build Structure step to proceed to the mode that allows selection of template regions for building.

Slide Freely (Edit menu): Use this button to edit the alignment between query and template. Selecting a residue and sliding to the right creates N-terminal gaps. Selecting a residue and sliding to the left creates C-terminal gaps.

Slide As Block (Edit menu): Use this button to edit the alignment between query and template. Selecting a residue and sliding to the right creates N-terminal gaps. Selecting a residue and sliding to the left slides all residues as a block to the left.

Add Anchors (Edit menu): Set alignment constraints by setting anchors at residue positions. To remove anchors, click on the same residue again.

Lock Gaps (Edit menu): To prevent gaps from being collapsed during manual editing, lock the gaps using this button. Locked gaps are indicated with a **-** symbol.

Unlock Gaps (Edit menu) To allow gaps to collapse during manual editing, unlock the gaps using this button. Unlocked gaps are indicated with a **~** symbol.

Color Scheme (Display menu): Select a color scheme for coloring sequences in the sequence viewer. If there is a structure in the 3D Workspace associated with the sequence, it will also be colored.

View Structure (Display menu): Use this button to display and undisplay structures in the Workspace.

View SSA (Display menu): Use this button to turn the secondary structure assignment (SSA) in the sequence viewer on and off.

2.5 Prime Job Options and Control

Before launching a Prime calculation from a step panel, you may want to choose which machine host the job will run on and check or modify job settings. When a run is complete, you can incorporate the results into the Maestro Project Table and save them as a project.

2.5.1 The Job Options Dialog Box

The Structure Prediction–Job Options dialog box, shown in Figure 2.2, allows you to specify which machine each type of Prime job will run on, and under which user name. To open the dialog box, choose Job Options from the File menu on the Prime menu bar. The Job Options dialog box lists 10 types of jobs, from Find Family (run from the Find Homologs step) to Run Refinement (run from the Refine Structure step). Specify the host machine and the user name for each type of job. The default host is `localhost`. Make sure your `schrodinger.hosts` file contains a list of available machines to run on. See the *Installation Guide* for information on the `schrodinger.hosts` file.

2.5.2 Setting Up and Launching Jobs

Though you specify the machines used to run jobs in the Job Options panel, you select Prime job settings and launch jobs from the Prime step panels. This can be as simple as clicking an action button, such as Search. In other steps, you select the type of job from a Task menu and then launch the job by clicking Run.

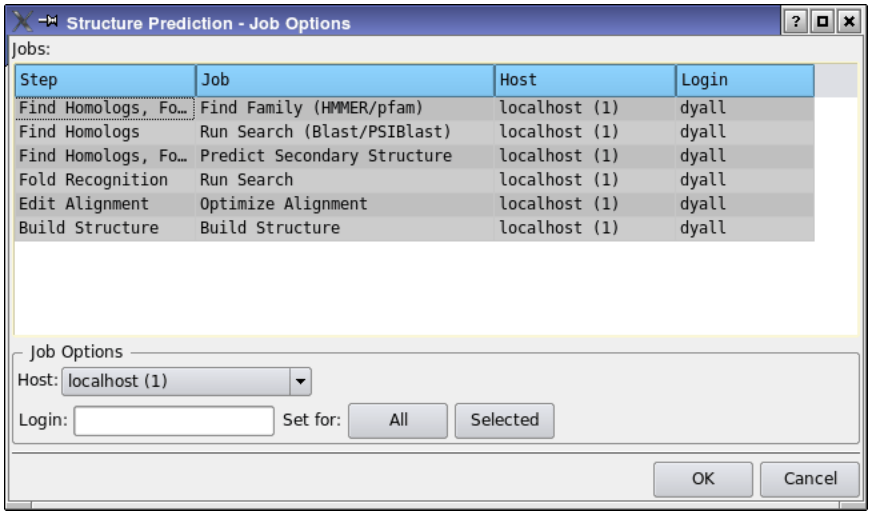

Figure 2.2. The Structure Prediction–Job Options dialog box.

You can modify settings for step actions or tasks using the Options button. You select the residues or secondary structure features on which a refinement task will operate from lists, tables, and the Atom Selection dialog box in the Refine Structure step panel. Until you make a selection, the Run button remains unavailable.

For more information on setting up and launching jobs, see the discussions of specific steps in the chapters that follow.

2.5.3 Monitoring Jobs

After you launch a job, the job status icon in the upper right corner of the step panel turns green and spins.

To monitor a job using the Monitor panel, click the job status icon. You can use this panel to monitor jobs from the current session or your previous sessions.

To kill a job, select the job name and then click Kill. If the job still appears to be running, you may have to go to the Project directory and remove the job file.

For more information about the Monitor panel and monitoring jobs, see Section 3.2 of the *Job Control Guide* and the online help.

2.5.4 Adding Structures to the Project Table

In the Build Structure step, predicted structures can be added to the Maestro Project Table as new entries. This is useful for comparing predicted structures produced from different runs, and for refining structures.

Prime–Structure Prediction: Initial Steps

This chapter outlines the initial steps in the Prime–SP workflow. In the Input Sequence step, a query sequence is imported. In the Find Homologs step, potential templates are identified, and a path is chosen: either Comparative Modeling or Threading. Later chapters describe the steps in each path in which a model is built.

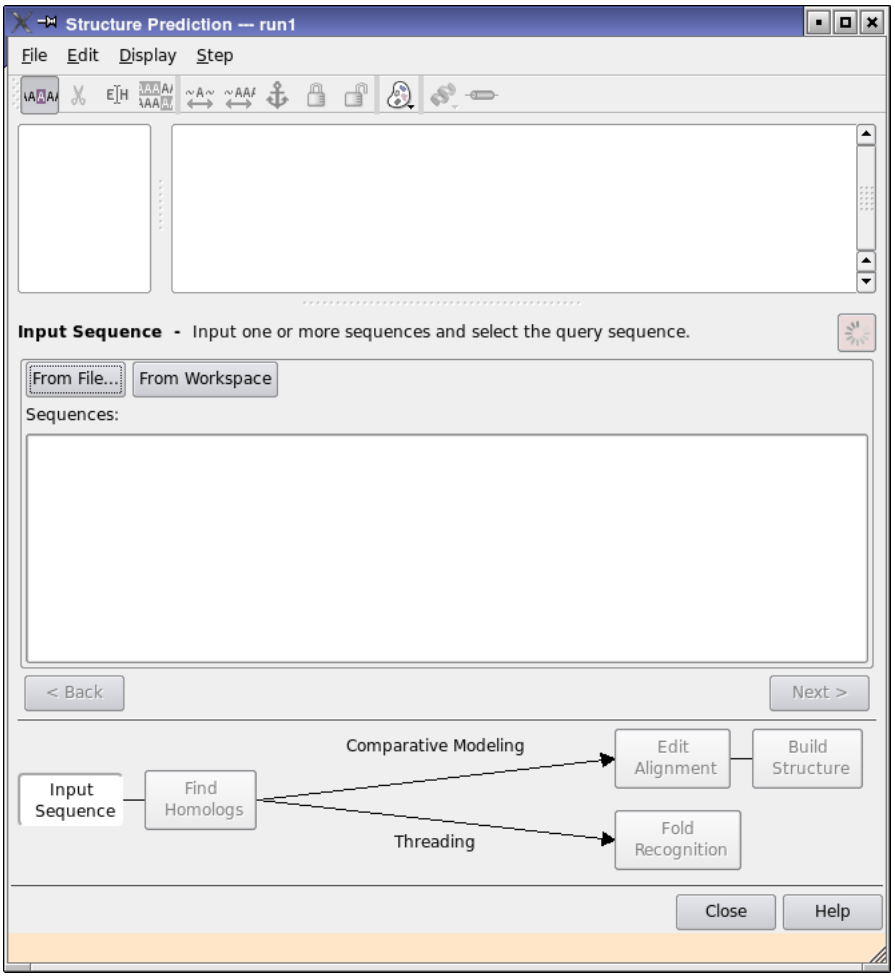

Figure 3.1. The Input Sequence step, initial view.

3.1 Input Sequence Step

This step is used to read in an amino acid sequence from a file or from a protein structure in the Workspace.

To read a sequence from a file:

- Click From File and specify a file containing the desired sequence.

  Several file formats are supported, including FASTA, EMBL, GENBANK, and PIR. For a complete list, see Appendix C. File formats are auto-detected by the program. Sequences must be in standard uppercase one-letter code.

To read a sequence from the Workspace:

- Click From Workspace to read sequences from proteins in the Workspace.

  If there are multiple chains with the same sequence, the sequence will be read in only once.

After input, the Sequences table in the Step View area shows all available unique sequences with their corresponding chain lengths in the order in which they are read. By default, the first sequence read in is displayed in the sequence viewer in the upper section of the panel. If multiple sequences have been read in, only one can be brought forward to the next step in a given Prime run.

To select a query from the Sequences table:

- Click on a sequence to select it for this run.

  This brings the selected sequence into the sequence viewer. The sequences are named by extracting the name from either the PDB header or the FASTA file header. Non-unique sequences are ignored. Non-unique names are modified to make them unique.

Note: Sequences longer than 1,000 residues generally include more than one domain. Divide these sequences into probable domains and treat each domain as an individual query.

To continue:

- When you are satisfied with the query sequence, click Next to go to the Find Homologs step.

  If an error message appears when you click Next, you cannot proceed until the problem is resolved. Error messages are listed in Appendix D. See Section D.1 on page 127 for help with Input Sequence error messages.

3.2 Find Homologs Step

This step is used to search for homologs of the query sequence. These homologs can then be used as templates to build a model structure.

3.2.1 Importing Homologs

To import a homolog not in the NCBI sequence database, click Import. This button is also used to import homologs that have been rotated into a common reference frame when multiple templates have been selected.

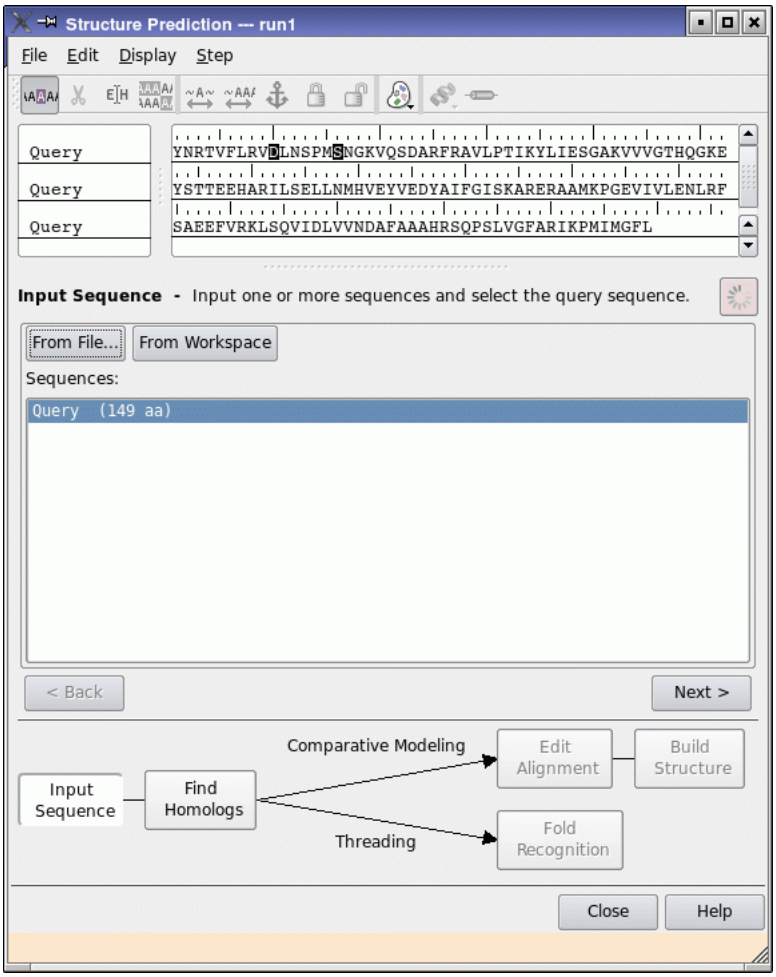

Figure 3.2. The Input Sequence step, after input.

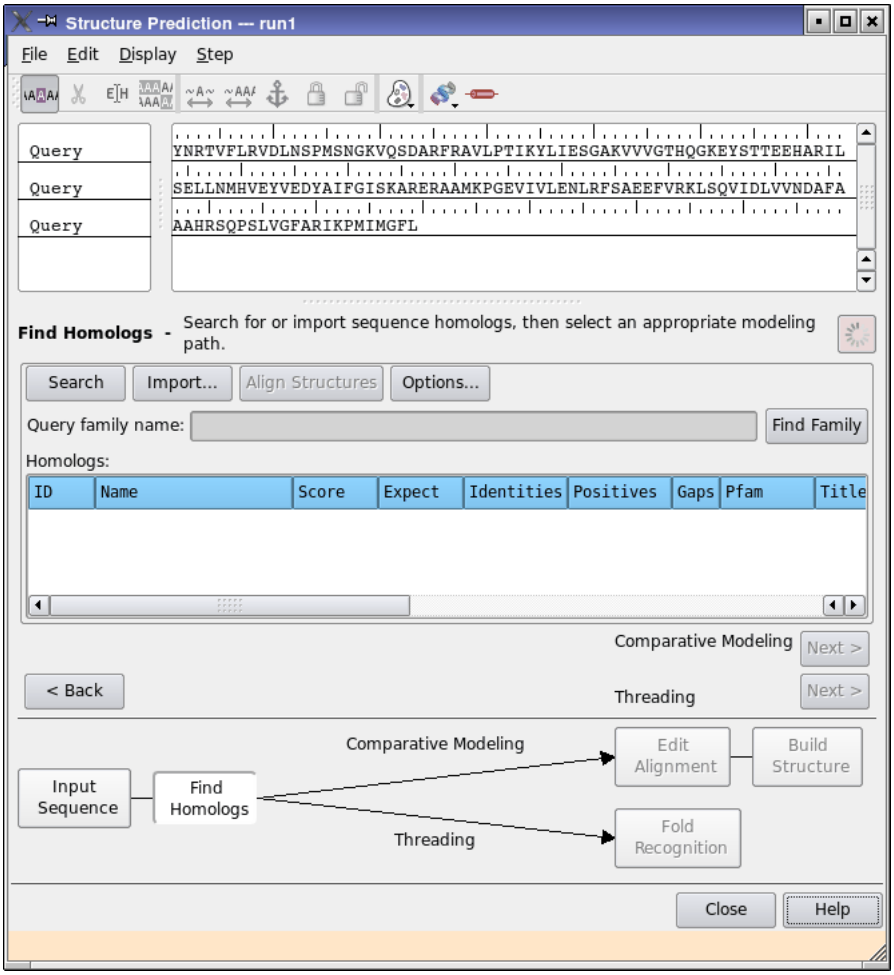

Figure 3.3. The Find Homologs step, initial view.

3.2.2 Searching for Homologs

To launch a search for homologs:

- Click Search (near the middle of the panel).

  Depending on what is selected in the Find Homologs–Options dialog box, BLAST or PSI-BLAST is used to search for templates. The default is to use BLAST to search the non-redundant PDB database that has been loaded with Prime. For search options, see Section 3.2.3.

To identify a family for the query (optional):

- Click Find Family to run HMMER/Pfam.

  This helps identify the family that best matches the query sequence. This job should take only a few minutes.

The query family name found appears in the Query family name text box with an E-value in parentheses. If the E-value of the family is too large, it is not meaningful to consider the family.

The sequence viewer shows the match between the query and the consensus sequence of the family. If this sequence, labeled *NAME*_pfam, is not visible, check for a plus sign to the left of the query name: *+NAME*. Click the plus sign to show any hidden sequences. The match sequence displays information about which residues are conserved in the multiple sequence alignment used to generate the Hidden Markov Model (HMM). Capital letters indicate highly conserved residues, lowercase letters indicate a match to the HMM, + means the match is conservative, and a blank indicates that the residue does not match the HMM.

3.2.3 Find Homologs Search Options

To change search options:

- Click Options to open the Find Homologs–Options dialog box.

You can use this dialog box to specify a PSI-BLAST or BLAST search, or change the settings for either type of search.

General options include:

- Database
- Similarity Matrix
- Gap Costs
- Expectation value
- Word size
- Filter query (off by default)

PSI-Blast options include:

- Inclusion threshold
- Number of iterations

The default database selection is NCBI PDB (all). Other selections include:

- NCBI PDB (non-redundant)—Does not include PDB files with identical sequences.
- NCBI NR—Recommended for PSI-BLAST runs. With this selection, only the sequences with structures available in the PDB are displayed in the Homologs table.

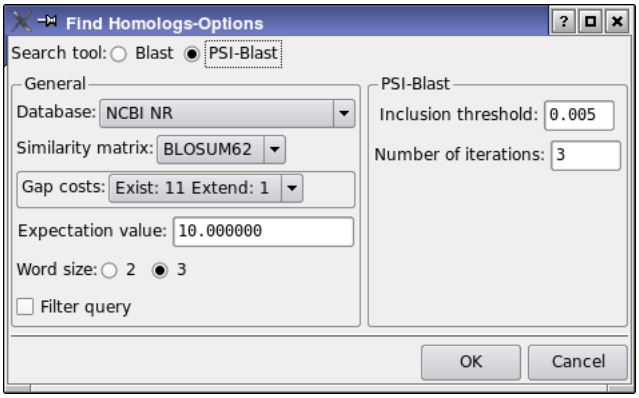

Figure 3.4. The Find Homologs–Options dialog box.

3.2.4 Search Results

The output from the homolog search appears in the table labeled Homologs: (*N* found, *n* selected). For each potential template, the row contains the ID, the name, score, expectation value, % identities, % positives, % gaps, and header information (if taken from a PDB file). All percentages are rounded to the nearest whole number, which may be zero. A description of the table columns is given in Table 3.1.

You can view the complete text of each table cell as a tooltip by pausing the pointer over the cell. Columns can be sorted by clicking on the column heading. Click once to sort in descending order, and click twice to sort in ascending order. The column widths can be adjusted using the left mouse button to drag the column border.

If no structural coordinates are available for a template, that row is dimmed and cannot be selected.

If a chain ID is given for a homolog (e.g., Chain A in `1EMS_A`), then that chain, not the entire protein, is the potential template. When aligning multiple templates with multiple chains, you may need to extract single chains using the `getpdb` utility, as described in Section D.2.3 of the *Maestro User Manual*.

Once homologs are found, click on a row to view the structure of that template in the Maestro workspace. If no structural coordinates are available for selection, the row is dimmed. Selecting a template will also bring the sequence alignment of the template and the query sequence into the sequence viewer. Click on another template to deselect the first one and display the selection in the Workspace window.

After you review the alignments, select the template that you want to build on by clicking on its row in the table. To select multiple templates, see Section 3.2.5.

Table 3.1. Homologs table data

Column Header	Description
ID	The PDB ID code, including chain name. If a chain name is specified, then the chain, not the entire protein, is the potential template.
Name	Sequence name provided by BLAST.
Score	BLAST bit score.
Expect	BLAST expectation value.
Identities (%)	Percentage of residues that are identical between the sequences.
Positives (%)	Percentage of residues that are positive matches according to the similarity matrix selected in the Options dialog (default matrix is BLOSUM62).
Gaps (%)	Percentage of gaps in both query and homolog as returned by BLAST.
Title	From the PDB TITLE field.
Compound	From the PDB COMPND field.
Source	From the PDB SOURCE field.
Experiment	From the PDB EXPDTA field. This gives the experimental technique used to obtain the structure, e.g., X-RAY DIFFRACTION or NMR.
Resolution	From the PDB REMARK2 record, where applicable.
Ligands/Cofactors	From the PDB HET records: hetID (three alphanumeric characters) and name for each HET group.
Warning	A warning appears in this column for sequences that may be unusable for model building. See Section D.2.2, for a list of warnings.

3.2.5 Multiple Templates and Structure Alignment

To select more than one template for building, hold down the CTRL key and select another homolog in the table, or hold down the SHIFT key and select a set of homologs. If more than one template is selected, they will need to be *structurally aligned*, that is, rotated to bring them into a common reference frame, before they can be used in the next step.

- If you have already created a file with structurally aligned multiple templates, import it using the Import button, as described in Section 3.2.1.
- If you have not yet performed structural alignment on the selected templates, click the Align Structures button.

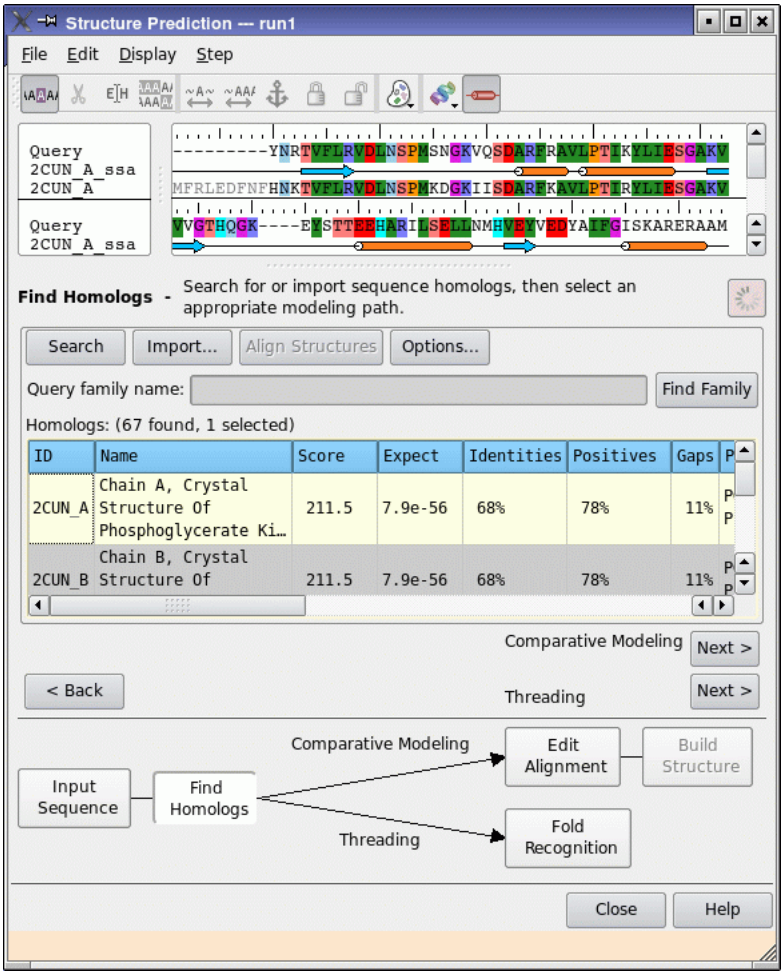

Figure 3.5. The Find Homologs step with search results.

If structure alignment fails or does not give the intended result, the problem may be one of the following:

- The selected templates are structurally too dissimilar for a meaningful alignment.
- One or more of the template structures includes multiple chains, and either
  - Two or more templates are chains from one `.pdb` file (the structure alignment facility will not align chains from the same file), or
  - The chains that were aligned were not the chains you intended to align.

If the problem involves a template with multiple chains, you can use the `getpdb` utility (see Section D.2.3 of the *Maestro User Manual*) to extract each chain into a separate `.pdb` file, then run structure alignment again.

For information on running structure alignment from the command line, see Section 12.1.

3.2.6 Continuing to the Next Step

When you are satisfied with the template (or structurally aligned multiple templates) you have selected, follow the Comparative Modeling Path by proceeding to the Edit Alignment step.

If no satisfactory templates can be identified by BLAST/PSI-BLAST and you have not imported a template, you can follow the Threading Path by proceeding to the Fold Recognition step.

Note: The maximum length of a query in the Threading Path is 1,000 residues. It is recommended that such long sequences be divided into queries by probable domains, if this has not been done already, and the homolog search be repeated for these shorter queries before entering the Threading Path.

3.2.7 Saving Runs With Different Templates

If you decide later to try a different template, you can avoid overwriting your existing workflow by starting a new one before you navigate back to Find Homologs to choose a different homolog. Choose Save As from the File menu. This creates a copy of the current run, allowing you to select a new template and build on it. You can then compare results from each run.

3.2.8 Error Messages

If there is an error, a message appears when you click Next to go to the next step. See Section D.2 on page 127 for a list of error messages for Find Homologs.

Comparative Modeling: Edit Alignment

4.1 The Comparative Modeling Path

In the initial steps of Prime–SP, you imported a query sequence (Input Sequence) and identified possible templates (Find Homologs). If a query-template match with a substantial percentage of identical residues has been selected, it is appropriate to continue the structure prediction process by following the Comparative Modeling path, a series of two Prime–SP steps used to build a model structure of the query sequence based on homology to one or more templates.

The structure prediction process in the Comparative Modeling path includes the following steps:

1. A satisfactory query-templates sequence alignment is produced in Edit Alignment.
2. The selected query-template alignment is used to build a model structure of the query in Build Structure.

4.2 Overview of the Edit Alignment Step

In order to build a model structure of the query, the Build Structure step requires a satisfactory alignment between templates and query. The alignment of templates to query generated in the Find Homologs step is based only on sequence. Therefore, there is room for improvement. The Edit Alignment step allows you to improve alignments using various tools before building a model in the Build Structure step.

There are several ways to produce an alignment for use in Build Structure:

- Accept the BLAST/PSI-BLAST alignment generated in the Find Homologs step and currently displayed in the sequence viewer.
- Import an alignment.
- Generate or import secondary structure predictions, then use the Align program to create a new alignment.
- Edit any of these alignments manually.

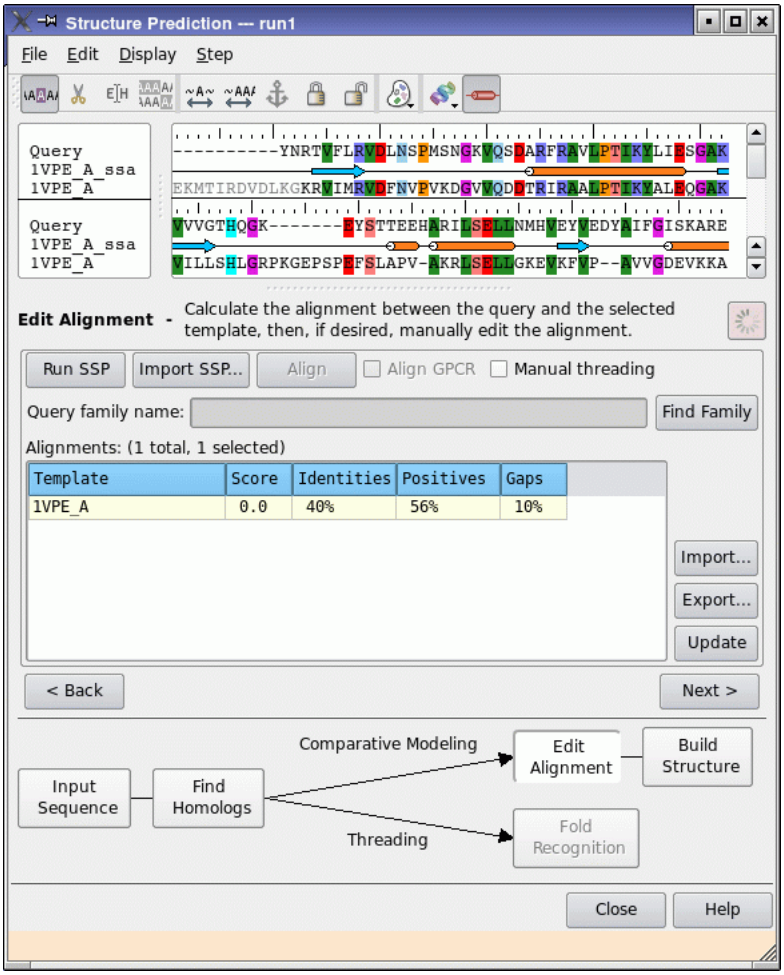

Figure 4.1. The Edit Alignment panel, initial view.

If you want to search for a family that best matches the query sequence, but did not do so in the Find Homologs step, click Find Family to run HMMER/Pfam. For more information on this option, see Section 3.2.2 on page 20.

When you are satisfied with the alignment, click the Next button to proceed to the Build Structure step.

4.3 Initial and Imported Alignments

4.3.1 Accepting the BLAST/PSI-BLAST Alignment

When you start the Edit Alignment step, the BLAST/PSI-BLAST alignment for the template you selected in Find Homologs is displayed in the sequence viewer. See Figure 4.1.

If this is the template and the alignment you want to use, click Next and go to Build Structure. To accept a different template, using its existing alignment, click on the desired template and then click Next.

4.3.2 Exporting an Alignment

You may want to save the existing alignment for later use before making changes. Click the Export button to the right of the Alignments table to open the Export Alignments To File panel. To import the alignment again, see Section 4.3.3.

4.3.3 Importing an Alignment

Prime can also use an alignment generated in another program (or exported previously from Prime). The sequences in the alignment file must be exactly the same as those used by Prime, and so must the sequence and structure (PDB) names.

Note: The template PDB ID must be uppercase in the alignment file.

To import an alignment, click the Import button to the right of the Alignments table. This opens the file selection window (Import Alignments From File). For details of the allowed formats, see Appendix C.

4.4 Secondary Structure Predictions (SSPs)

4.4.1 Generating SSPs

A secondary structure prediction for the query is required to run the Align program. To run all available SSP programs, click Run SSP. One secondary structure prediction program (SSpro) is bundled with Prime. However, the optional third-party program PSIPRED is highly recommended for optimal results, especially for GPCRs. PSIPRED is not available on Windows. See Appendix A for more information on this program.

When the SSP run is finished, the secondary structure predictions are displayed in the sequence viewer below the query sequence. If there is a problem running the structure prediction programs, an error message is displayed.

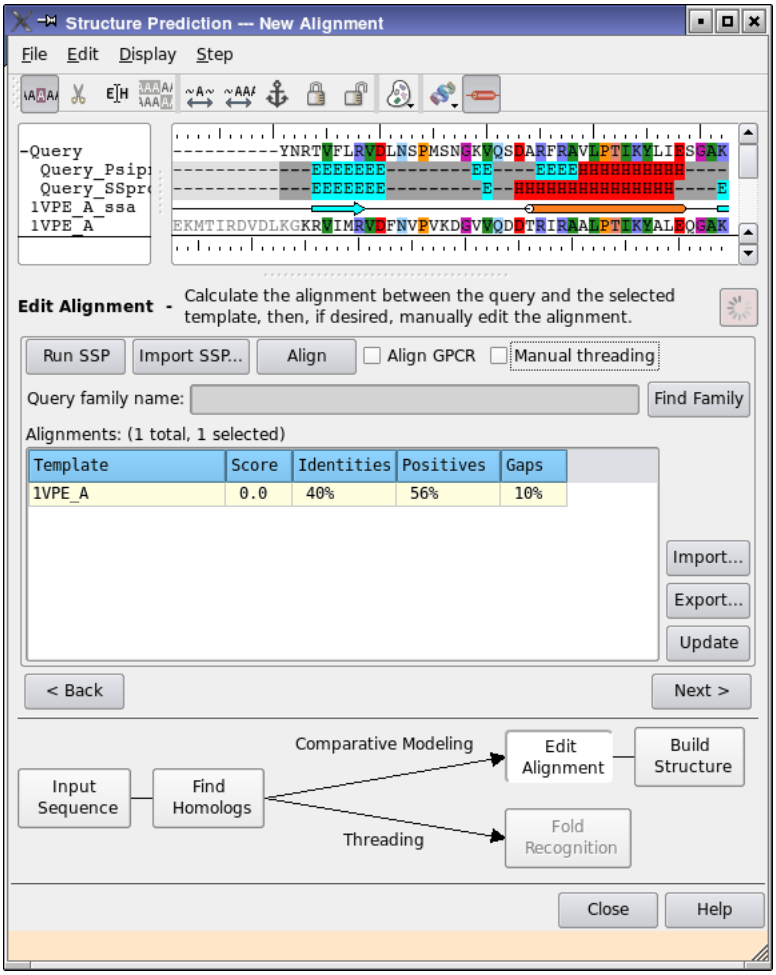

Figure 4.2. The Edit Alignment step, after Run SSP.

4.4.2 Exporting and Importing SSPs

You may want to save an SSP for later use. Right-click on an SSP for a menu of options, including deleting the SSP, hiding all SSPs, or opening the Export SSP panel. By default, the SSP is saved in native Schrödinger format. If there is a problem exporting a file, an error message is displayed.

To import a secondary structure prediction for the query, click Import SSP and select the SSP file, which may be in FASTA or Maestro format. If necessary, use the `seqconvert` utility to change the file format. (See Section 12.2 for more information on `seqconvert`.) If the file

contains bad data or the sequence doesn't match the query sequence, an error message is displayed.

4.4.3 Deleting or Editing an SSP

If you believe an SSP is incorrect, you may delete it entirely or edit the incorrect portions. To delete an SSP, right-click on the SSP and choose Delete from the shortcut menu.

To edit an SSP:

1. Click the Edit SSP toolbar button:

   

2. Highlight the part of the SSP you want to change.
3. Type the character `e`, `h`, or `-`.

   E represents strand, H represents helix, and – represents loop.

The highlighted region of the prediction is changed to your choice.

4.4.4 Resetting SSPs

To retrieve an unmodified SSP after editing, click Run SSP. This immediately resets the SSP, but does not rerun the prediction programs.

4.5 Generated and Modified Alignments

4.5.1 Running the Align Program

Before running an alignment, if you want to take advantage of the special capabilities for aligning GPCRs, select Align GPCRs. These capabilities include fingerprint matching, a customized GPCR sequence database, and identification of transmembrane helices.

To take advantage of expert knowledge, one or more pairs of query/template residues that you know should be aligned can be constrained to remain so, by using alignment constraints.

To constrain a pair of residues to remain aligned:

1. Click the Add Anchors toolbar button.

   

2. Select one of the residues in the pair that you want to remain aligned during the Align calculation.

   An anchor symbol is displayed in the ruler at the corresponding residue position.

When you are satisfied with the secondary structure predictions for the query, click Align to start the alignment job.

When Align completes the generation of the new alignment, you may accept it and continue to the next step or edit it manually as described below.

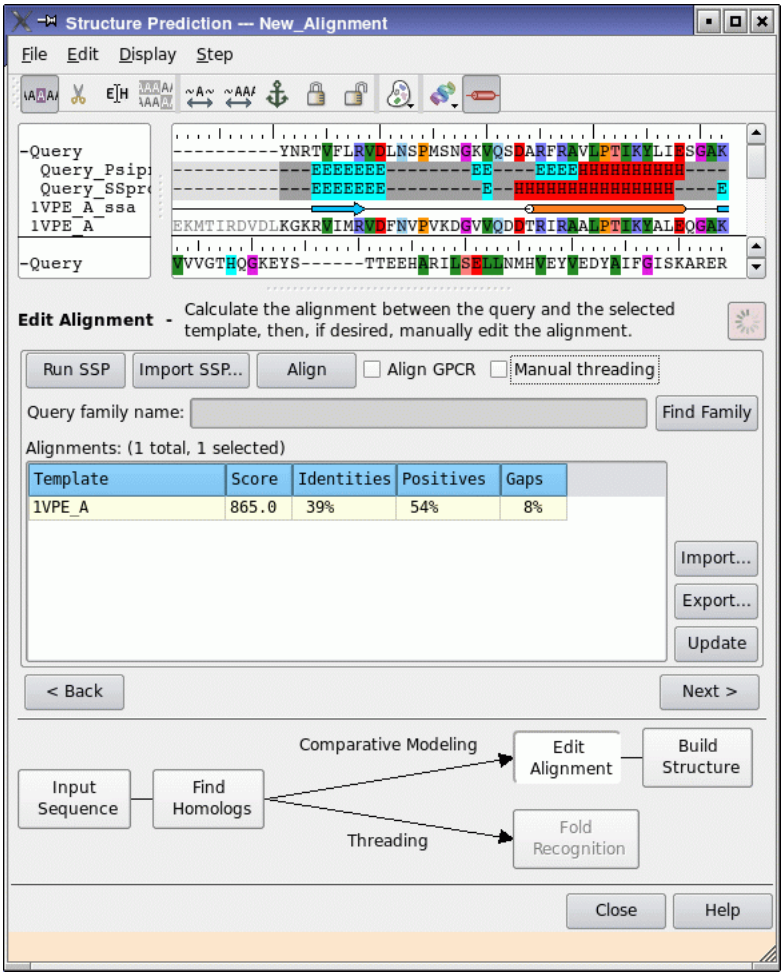

Figure 4.3. The Edit Alignment step after alignment.

4.5.2 Align Program Technical Details

The goal of the program launched from the Edit Alignment step, Single Template Alignment, is to generate an accurate alignment between proteins with medium to high sequence identity (20-90%). Features of the program include:

- A position-specific substitutional matrix (PSSM) for the query sequence, derived from PSI-BLAST, is used to match the template sequence.
- In order to minimize the inaccuracy in a single secondary structure prediction, a novel and robust algorithm has been designed to derive a composite secondary structure for the query sequence from multiple predictions. The composite SSP is aligned to the SSA of the template.

4.5.3 Manually Editing the Alignment

You can edit any alignment, however obtained, using combinations of these Prime toolbar buttons: Slide Freely, Slide as Block, Add(/Remove) Anchors, Lock Gaps, and Unlock Gaps.

Slide Freely (Edit menu): Edit the alignment between query and template. Selecting a residue and sliding to the right creates N-terminal gaps. Selecting a residue and sliding to the left creates C-terminal gaps.

Slide As Block (Edit menu): Edit the alignment between query and template. Selecting a residue and sliding to the right slides all residues as a block to the right. Selecting a residue and sliding to the left slides all residues as a block to the left.

Add Anchors (Edit menu): Set alignment constraints by setting anchors at residue positions. To remove anchors, click on the same residue again.

Lock Gaps (Edit menu): To prevent gaps from being collapsed during manual editing, lock the gaps using this button. Locked gaps are indicated with a **-** symbol.

Unlock Gaps (Edit menu): To allow gaps to collapse during manual editing, unlock the gaps using this button. Unlocked gaps are indicated with a **~** symbol.

To assist in this task, you can view a preliminary model of the query in the Workspace. Select Manual Threading and click Update. The Workspace shows the template structure colored by query residue property. Each template residue is colored according to the property of the corresponding query residue in the alignment, revealing the location of charged, polar, and hydrophobic query residues. Use this option to make sure that insertions/gaps are not in the middle of regions of secondary structure such as alpha helices or beta sheets. It can also be used to check that hydrophobic residues point into hydrophobic regions and polar residues point

towards solvent. As you make changes to the alignment, click Update again to see how each change has altered the mapping of the query's residue properties onto the template structure.

If there are obvious errors in the alignment (such as gaps in regions of secondary structure), a warning will appear in the Build Structure dialog box. Manual editing can be used to adjust the alignment before attempting to build the structure again.

4.5.4 Updating the Step View Table

After the Align job is complete, the Step View table is updated with the new alignment score, the percent identity, the percent similarity, and the percent gaps (all rounded to the nearest integer). This information can be updated after any change, including manual editing of the alignment, by clicking the Update button. The changed alignment is also reflected in the Maestro Workspace display of the structure.

4.6 Error Messages

When you click Next at the end of the Edit Alignment step, the Check Alignment warning may appear. The warning lists possible alignment problems that you may want to fix before continuing to the next step. For example, gaps in secondary structure elements of the template, or gaps in the query that are aligned to secondary structure elements of the template.

If you do not want to make corrections to the alignment, click Continue to go to the next step. To make corrections based on the information in the error message, make a note of the gap locations given, click Cancel to close the message box while remaining in Edit Alignment, and change the alignment accordingly.

Comparative Modeling: Build Structure

5.1 The Build Process

The Build Structure step builds a model structure of the query sequence based on the templates selected in the previous step. Figure 5.1 shows the initial view of the Build Structure step.

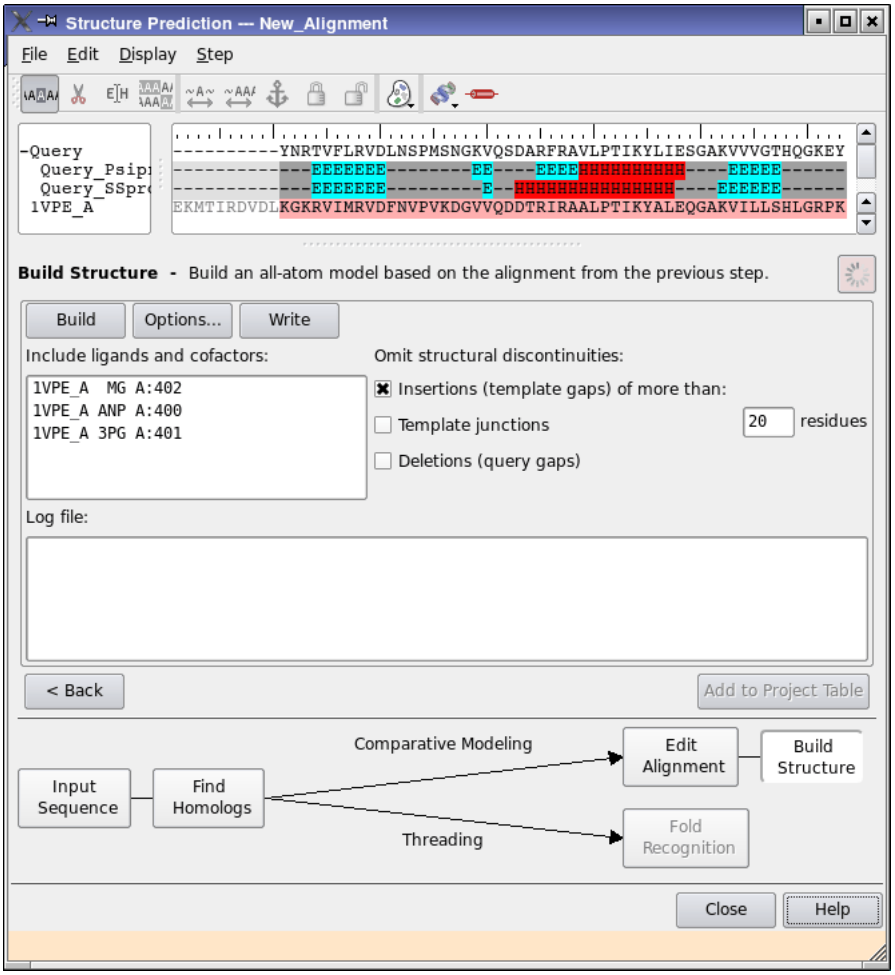

Figure 5.1. The Build Structure step, initial view.

The Build process includes the following steps:

1. Coordination of the copying of backbone atoms for aligned regions and side chains of conserved residues
2. Building insertions and closing deletions in the alignment
3. Building tails out to the last residue in the template

The Build process may also include the following steps:

- Side-chain prediction of non-conserved residues
- Minimization of all atoms not derived directly from the template

You can specify which optional steps to perform by clicking Options and making selections in the dialog box. See Section 5.2.3.

5.2 Preparing to Build

If you want to build a model using multiple templates or including ligands or cofactors, you must specify how these are to be treated before clicking the Build button to start the building process.

5.2.1 Multiple Templates (Set Template Regions)

If you intend to use multiple templates to build a model of the query, you will have already selected them and ensured that they are structurally aligned (i.e., rotated into a common coordinate frame; see Section 3.2.5). In the Build Structure step, before starting the structure build by clicking the Build button, you must specify which template to use for each region of the query. If you do not select regions from other templates, the topmost template in the table will be used for the entire query.

To enter the mode that allows you to set the template regions, click the Set Template Regions toolbar button:

Highlight the region of each template in the sequence viewer that you want to use. Click the Set Template Regions button again to end the selection. (You may need to scroll over and select again if the desired region is wider than the display window.)

The Set Template Regions tool helps make sure that only one template residue is assigned to a query residue at one time. Regions of a template not being used to model the query are colored dark gray (regardless of coloring schemes). When you begin to set template regions, only the

first template is in use. As you select regions in the second template, the tool automatically deselects the corresponding residues in the first template. If you select regions for use from a third template, the corresponding regions of the first and second templates are deselected, and so on.

5.2.2 Including Ligands and Cofactors

Any cofactors and ligands present in the template you have chosen (except HOH) are listed in the Include ligands and cofactors box. Click in the list to select one or more ligands and cofactors. Your selections are highlighted in yellow in the Workspace.

5.2.3 Build Options

Click Options to open the Build Structure–Options dialog box and specify which optional model building steps to perform:

- Retain rotamers for conserved residues—retain side-chain rotamers for conserved residues.
- Optimize side chains—Optimize the side chains.
- Minimize residues not derived from templates—minimize residues that are not derived from the templates. Only available if Optimize side chains has been selected.

By default, all three of the above options are selected.

5.2.4 Omitting Structural Discontinuities

You can disable the building of structural discontinuities arising from insertions, template junctions, and deletions. The discontinuities are capped with NMA and ACE, so the final structure is discontinuous. If you choose any of these options, you should check that your final structure is reasonable.

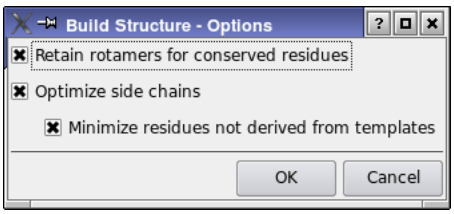

Figure 5.2. The Build Structure–Options dialog box.

If there is an insertion in the query (a template gap) that requires the building of a long loop, you can instead cut the query sequence and cap it with NMA and ACE, and not build the loop, by selecting the Insertions (template gaps) option under Omit structural discontinuities, and enter the maximum length of loops that will be built in the text box. This option could be useful where there are long insertions that are not in the region of interest.

Likewise, if there are deletions in the query (a query gap), or if there is a junction between two templates, you can choose to cap the ends of the two pieces and not try to join them in building the structure. To do this, select Deletions (query gaps) or Template junctions under Omit structural discontinuities.

When the structure is built with discontinuities, the results appear to be misaligned in the sequence viewer. To display the correct alignment, right-click in the sequence viewer and choose Align by Residue Number.

5.3 Running the Build Structure Job

When you click Build, the model-building program begins, and its log is displayed in real time in the Log file window (see Figure 5.3). It can also be viewed in the Monitor panel (click the job status icon in the upper right corner of any Prime–SP panel to open the Monitor panel). Finally, the build log is saved to a log file in the directory where Prime is running.

First, the log lists the chosen templates. If you did not intend to choose more than one template, or if you neglected to align the templates, stop the job and restart it with the correct template. Jobs can be stopped, paused, and resumed from the Monitor panel. See Section 2.5 on page 14 for more information on Job Control.

The next part of the log file gives information about the build process for template transition regions, then for insertions, and then for side chains, followed by non-template regions. The last line of the log describes the diagnosis and success of model building.

Once the model building calculations are complete, the model is displayed in the Workspace superimposed on the template, using the color scheme Atom PDB B Factor (Temperature Factor). Blue atoms are those derived directly from the template; red atoms have been predicted or modeled. To restore this color scheme, choose Atom PDB B Factor (Temperature Factor) from the Color all atoms by scheme toolbar button menu in the main window.

Build Structure also creates an atom set named non_template_residues which can be found in the Sets tab of the Atom Selection dialog box. This set of inserted residues is generally chosen to undergo refinement.

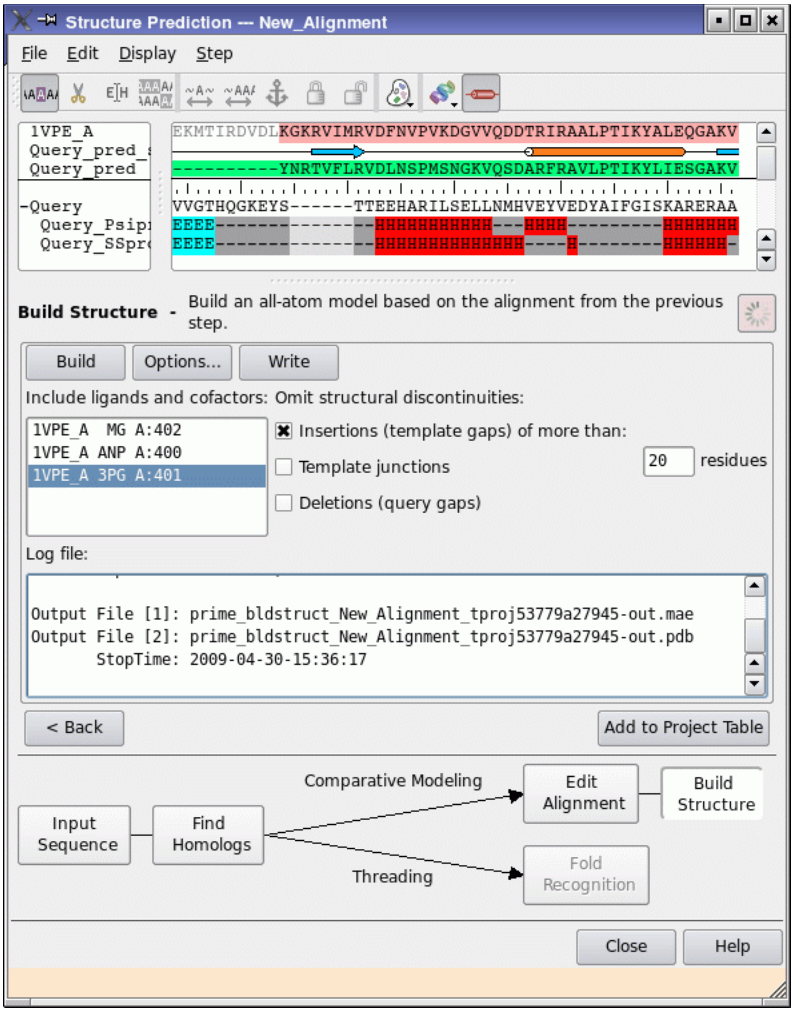

Figure 5.3. The Build Structure step after building.

The sequence viewer contains the query sequence, the templates, and the sequence of the newly predicted structure in that order. To view only the model structure, select View Structure from the Display menu and then select Predicted Only. (The other options are All and None.)

You can also run the Build Structure job from the command line. To do so, click Write to write the input file. The file is written to the current working directory, and is named according to the run and project. A dialog box informs you of the file name and the command to use, which is:

`$SCHRODINGER/bldstruct` *input-file*

For more information on this command, see Section 11.2 on page 88.

5.4 Saving and Exporting Built Structures

When building is complete, the built structures can be added to the Maestro project by clicking Add to Project Table. Once you have added the structure to the Project Table, you can refine the structure by choosing Applications > Prime > Refinement in the main window. You can also export any Project Table entry, such as a built structure, to a file of another format. For information on exporting structures, see Section 3.2 of the *Maestro User Manual*.

5.5 Technical Notes for the Build Structure Step

The Build Structure step uses the following resources and methods:

- The OPLS_2005 all-atom force field for energy scoring of proteins as well as for ligands and other non-amino-acid residues.
- A Surface Generalized Born (SGB) continuum solvation model for treating solvation energies and effects.
- Residue-specific side-chain rotamer and backbone dihedral libraries, derived from the non-redundant data sets extracted from the PDB, and representing values most commonly observed in protein crystal structures. The libraries allow for both the evaluation of existing rotamer/dihedral values and the systematic prediction of new ones.

Both Build Structure and the step that follows, Refine Structure, can treat the 20 standard amino acids as well as modified residues and ligands.

Model building uses the following procedure:

1. Non-conserved residues are mutated to the desired identity. Side chains are added by finding the first rotamer in the library that does not produce a clash.
2. Insertions, deletions, and template transitions are built. These are cases in which the backbone itself needs to be reconstructed, either due to gaps in the alignment or template transitions that produce gaps in the 3D structure. These gaps are closed by reconstruction of the affected region ab initio, using a backbone dihedral library. If no reasonable conformation can be generated that closes the loop structure, the region being reconstructed will be expanded until closure is possible.
3. Gap reconstruction is done by finding any single loop conformation that closes the structure and is physically reasonable. If multiple conformations are found, the one that deviates from the template structure as little as possible is chosen. No attempt is made to perform an exhaustive optimization of the region.

4. Side-chain prediction (optional). Side chains may be optimized. The two choices are either: (a) optimize all side chains from scratch, or (b) optimize only those that are not in the template.
5. Minimization of non-template regions (optional). Any portions of the backbone that were not derived directly from one of the templates (and were thus reconstructed ab initio) are subject to a local minimization as described in Section 7.7 on page 62.

Fold Recognition

When sequence searches in the Find Homologs step of Prime–SP fail to find templates, or query-template sequence identity is low, structure prediction may be continued using the Threading path. The Threading path involves use of Fold Recognition to find candidate "seed" templates, followed by a return to the Comparative Modeling path with a selection of templates. Fold recognition is not available on Windows.

6.1 Overview of the Fold Recognition Step

The Fold Recognition step is designed to find structural templates that could not be found by a sequence search. The use of secondary structure matching is important in finding remote (<15%) homologs. Fold Recognition may identify a family or fold type for an unknown sequence.

The Fold Recognition process begins with the production of a secondary structure prediction profile for the query sequence. Once the query's SSP profile is generated, it can be used to identify candidate templates of known structure. Preliminary models of the alignment of the query to potential templates are ranked and displayed in the Templates table. The best of these preliminary models are selected as "seed" templates and can be evaluated for suitability to build a structure with comparative modeling.

6.2 Producing an SSP Profile

Secondary structure predictions to be used in the query's SSP profile may either be imported from a file or generated within Prime. The SSPs can then be exported, deleted, or edited to produce a satisfactory SSP profile for the query.

6.2.1 Importing and Exporting SSPs

To import a secondary structure prediction for the query SSP profile, click the Import SSP button and select an SSP file. Files may be FASTA or `m2io` (native Schrödinger) format. If necessary, use the `seqconvert` utility to change the file format. (See Section 12.2 for more information on `seqconvert`.) If the file contains bad data or the sequence doesn't match the query sequence, an error message is displayed.

To export an SSP, right-click on the SSP in the sequence viewer and select Export.

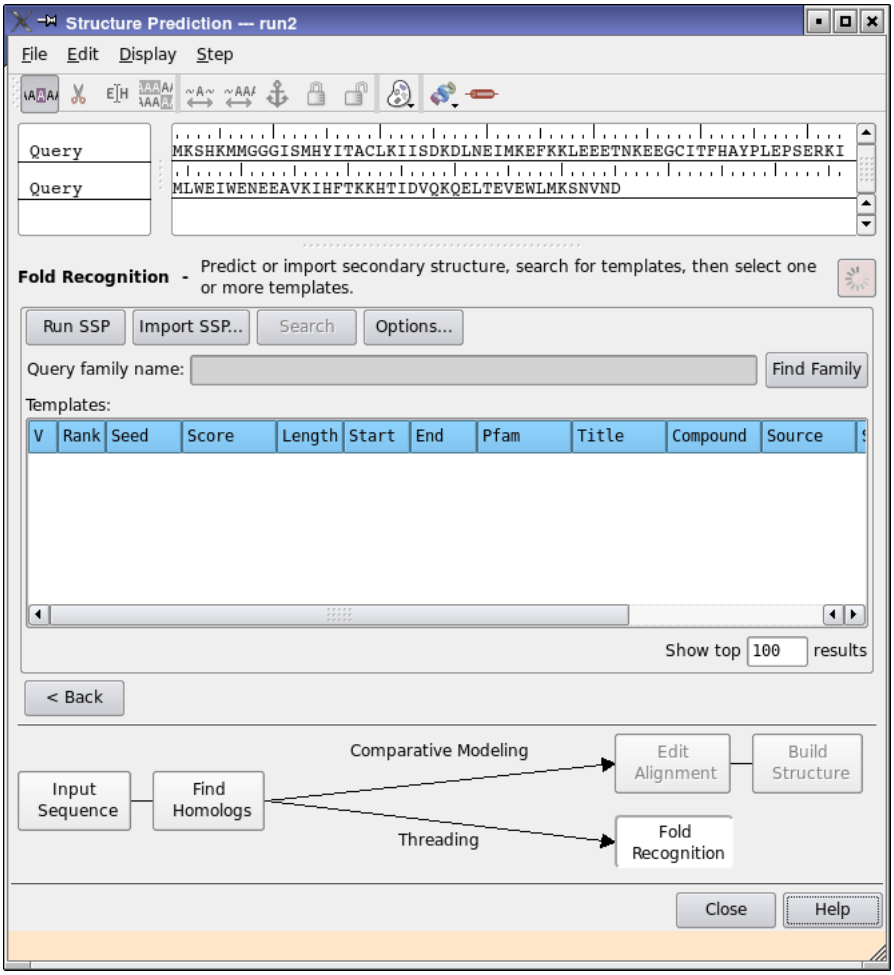

Figure 6.1. The Fold Recognition panel, initial view.

6.2.2 Generating SSPs

To generate SSPs for the SSP profile, run all available SSP programs by clicking the Run SSP button. One secondary structure prediction program (SSpro) is bundled with Prime. However, an optional third party program is recommended for optimal results. See Chapter A for more information on this program.

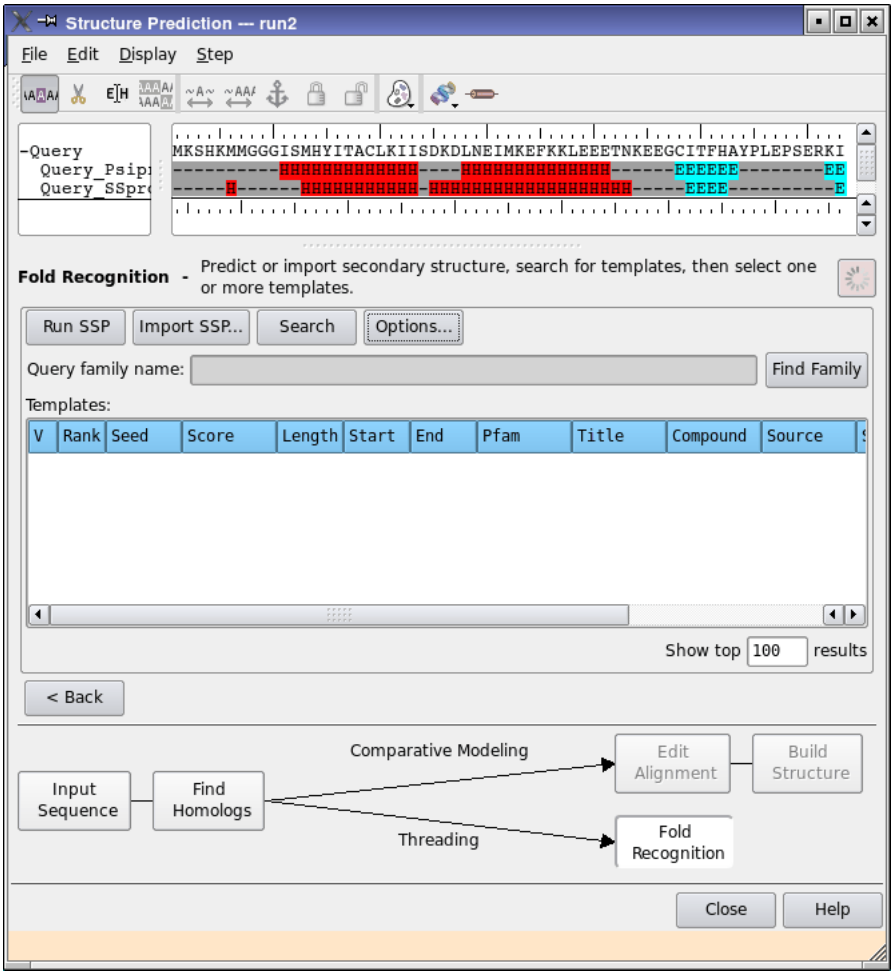

Figure 6.2. The Fold Recognition panel after Run SSP.

When the secondary structure prediction jobs finish, the SSPs are displayed in the sequence viewer below (as children of) the query sequence. (If an error occurs when running a structure prediction program, an error message is printed to the log file displayed in the Monitor panel.)

6.2.3 Editing or Deleting SSPs

You can use your expert knowledge to improve the query's SSP profile by deleting or editing SSPs you believe are incorrect.

To delete an SSP, right-click on the SSP in the sequence viewer and choose Delete from the shortcut menu.

To edit an SSP:

1. Click the Edit SSP toolbar button:

   

2. Highlight the part of the SSP you want to change.
3. Type the character `e`, `h`, or `-`.

   E represents strand, H represents helix, and – represents loop.

The highlighted region of the prediction is changed to your choice.

6.2.4 Resetting SSPs

To retrieve an unmodified SSP after editing, click Run SSP. This immediately resets the SSP (but does not rerun the prediction programs).

6.3 Searching for Templates

When you are satisfied with the SSP profile for the query, click Search to run the Fold Recognition program. You may prefer to change the search options described in the following section before doing so.

6.3.1 Search Program Options

Display of Results

Show Top *N* Results

By default, the 100 top-ranked templates found are shown in the Templates table. This number can be changed in the Show Top *N* Results text box before clicking Search.

Z-Scoring

Searching can be done with or without Z-scoring. The Z-score shows how significant a match is relative to a random match. It is calculated by shuffling the query sequence. See Section 9.2 on page 71 for more information on Z-score filtering.

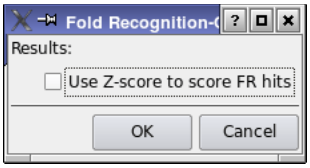

Figure 6.3. The Fold Recognition–Options dialog box.

You may want to turn on Z-scoring if the query protein is any of the following:

- Less than 120 residues long
- More than 90 percent alpha (helical)
- More than 90 percent beta (strand)

Click Options to open the Fold Recognition–Options dialog box and turn on Z-scoring.

Note: Running the search program with Z-scoring on a query with 300 residues typically requires three to four hours. Without Z-scoring, running a search on a protein of similar length typically takes only 10-15 minutes.

6.3.2 Search Program Technical Details

The search program finds potential templates for model building by searching a database of structure folds (SCOP domains) generated from the PDB using secondary structure information.

Profile-sequence matching and composite secondary structure information are the same as in the Align program used in Edit Alignment in the Comparative Modeling path. The search program used in the Threading path has two additional features:

- Weighting is adjusted by degree of structural conservation inside a family of templates. Each residue in the template sequence is defined as conserved or variable according to Multiple Structure Alignment (MSTRA) of the template with its structural neighbors. When aligning conserved residues, higher weight is given for matching and higher penalty is given for opening gaps, and vice versa for variable residues.
- Optional Z-scoring, as described above and discussed in more detail in Section 9.2 on page 71.

6.3.3 Search Output

When the search is complete, the Templates table displays candidate templates, ranked from the closest homolog to the least close. This preliminary ranking is based on a scoring function that combines sequence, secondary structure, and degree of structural conservation. If Z-scoring is on when Search is launched, rankings take Z-score into account as well.

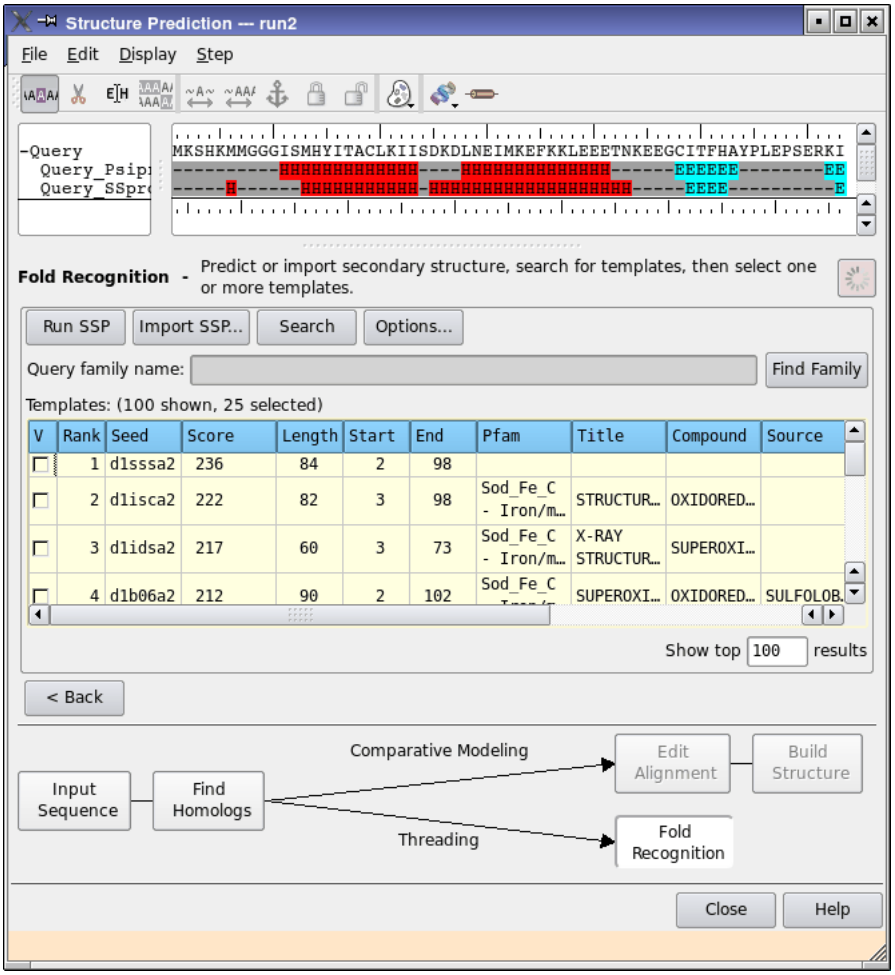

Figure 6.4. The Fold Recognition panel with Search results.

While it is possible that the top-ranked template will yield good results, it is recommended that multiple templates be used in model building. By default, the top 25 templates are selected. You can select the desired templates using control-click and shift-click. To sort the templates by properties other than rank, click the appropriate column heading.

Optionally, if you have not already searched for the family of the query, you can click Find Family to run HMMER/Pfam. The query family name found, with an E-value in parentheses, appears in the Query family name text box. The sequence viewer shows the match between the query and the consensus sequence of the family, as described in Section 3.2.2 on page 20. If a

potentially meaningful (E-value under 1) family identification is found, it may help you select different or additional seed templates to bring forward to Build Backbone.

To view a template colored by secondary structure, click the check box in the V (Visualization) column. The template appears in the sequence viewer, and its secondary structure can be compared to the SSP profile of the query.

Note: The query-template "alignments" that can be visualized in this step are provided only as a guide in identifying seed templates. Gaps or other unfavorable features will not affect the quality of the models built using these templates.

6.4 Evaluating Templates

To make use of the seed templates to build a model, they must first be evaluated for their suitability for comparative modeling. The evaluation is done with the use of two scripts. The process involves running an alignment for each template, then running the script to do the evaluation.

The first step is to export the SSPs in CASP format (or copy them from the working directory so that they have a `.casp` extension). It is recommended that you use both SSPs.

For each template that you want to use, run the alignment script as follows:

```
$SCHRODINGER/run -FROM psp CMAlign.pl target template seqFile sspFiles
```

The arguments for this command are described in Table 6.1.

Table 6.1. Arguments for the CMAlign.pl script.

Argument	Description
target	Name of the query sequence.
template	Name of the template, as shown in the Search table of the Fold Recognition step, including any underscore characters.
seqFile	Target sequence file.
sspFiles	CASP formatted secondary structure prediction files (`.casp`), blank-separated.

The alignments are stored in the subdirectory `CMAlign`. Once you have run `CMAlign.pl` for each of the desired templates, you can run the evaluation script:

```
cd CMAlign
$SCHRODINGER/run -FROM psp checkAlign.pl target
```

The script provides a recommendation for each template, and provides some information on the alignment.

6.5 Building a Model

Based on the recommendations of the script, you can use the Comparative Modeling path to build a model on any of the suitable templates. To do so, the template must be exported and read back into Prime, which you can do with the following procedure:

1. In the Fold Recognition step, include the template in the Workspace (click the V column).
2. Click the Create entry from Workspace main toolbar button to create a project entry.

3. Export the project entry as a PDB file.
4. In the Structure Prediction panel, create a new run and read in the query sequence again.
5. Click Next.
6. In the Find Homolog step, import the saved template PDB file.

   There is no need to run Search because a template was manually imported.
7. Select the imported template, and follow the Comparative Modeling path (or click Edit Alignment in the Guide).
8. In the Edit Alignment step, click Import SSP and select the `.casp` SSP files that you used for the analysis.
9. Click Align to run the alignment.

From this point, you can continue to the Build Structure step as normal.

Prime–Refinement

Prime–Refinement is a module used to refine protein structures from the Maestro Workspace. The methods can be applied to protein structures from any source, including one built using the Prime–Structure Prediction workflow. It runs independently of Prime–SP from its own Maestro panel, the Refinement panel. The Refinement panel offers the following refinement protocols: loop refinement, side-chain prediction, minimization, and a single-point energy calculation at the current geometry of the model structure.

In many cases, Prime–Refinement is used on protein structures from sources other than Prime, but it can also be used with structures added to a project from the Build Structure step in the Prime–SP module. A model structure built in the Build Structure step may require further refinement. If there are insertions or deletions in the alignment between query and template, it is recommended that you perform a loop refinement, since insertions and deletions are most frequently found in loop regions. As another example, if you have added a water molecule to the binding site of the protein, you can use Prime–Refinement on the composite entry.

Prime–Refinement can also refine structures with covalently bound ligands. The ligands can be multiply connected, covalently bound to each other and to the protein. This capability allows the refinement of proteins with phosphorylated residues or attached sugars, for example.

Prime–Refinement has an implicit membrane model, in which the membrane is modeled by a slab of low dielectric constant in which the protein is immersed. The width of the slab and the orientation of the protein with respect to the slab can be adjusted.

7.1 Preparing Structures for Refinement

Prime–Refinement has a great deal of flexibility in the types of structures it can handle, and can fix many structural problems. For standard residues, Prime–Refinement can fix formal charges and bond orders, and correct disparities between the sequence and the structure. For example, if a residue has the coordinates of ALA but is called SER, the 3HB will be ignored and the OG and HG added during refinement. The standard residues are the 20 canonical amino acids; ACE and NMA; HOH; CYX (disulfide); and the acid/base variants ASH/AS1 (ASP), GLH/GL1 (GLU), ARN (ARG), LYN (LYS), HIE/HIP/HID (HIS), CYT (CYS), SRO (SER), TYO (TYR).

Prime–Refinement has certain conditions that must be met by the input structures for a job to be run successfully. Some of these conditions only apply to covalently bound ligands.

For all kinds of structures, the following condition must be met:

- The structure must be an all-atom structure.

  This condition applies to the protein and any ligand, waters or cofactors present in the structure: all hydrogens must be present.

  You can add hydrogens to a structure by displaying the structure in the Workspace and double-clicking the Add hydrogens button in the toolbar.

  

  However, for structures containing nonstandard residues, you must correct bond orders and formal charges first.

For structures with nonstandard residues, the following conditions must be met:

- No residue can consist of disconnected pieces without formal bonds.

  This means, among other things, that metals should not have formal bonds to the rest of the structure, and should be treated as ionic. For example, in hemes the Fe must be a separate residue with a +2 formal charge and no bonds, and the "bonded" N atoms must have a –1 formal charge.

- Bound residues must contain at least 3 atoms between bonds to other residues.

  This means that, for example, a dihedral angle cannot span more than 2 residues.

- Bond orders and formal charges must be corrected.

  For nonstandard residues the supplied structure will be used, and you must check for correct bond orders and formal charges.

When you rename residues, you should be aware that OPLS_2005 parameters are used for standard residues and nonstandard residues.

Proteins can also be used as ligands, and the same conditions apply. No special treatment is needed for two distinct chains connected by a disulfide or similar side-chain linkage. If the "ligand" chain has nonstandard terminal groups or is connected to the main chain by its backbone, two steps are required to make sure that the "ligand" chain is treated correctly:

1. Rename standard residues appearing in the ligand to nonstandard names.
2. Rename the atoms in inter-residue C–N bonds between renamed (ligand) residues. For example, change "_C__ "to "_C*_", where the underscores represent spaces. Other atoms can keep their PDB names; only one of C or N needs to be changed.

If your structure does not meet one or more of the conditions outlined above, you must fix it before you can successfully run a refinement. Some of the problems only become apparent when you have run a refinement job, so it can be a useful diagnostic procedure to run a Prime energy calculation first. This job is quick, and produces warnings in the log file, which you can check in the Monitor panel.

Many of the preparation tasks are performed automatically or interactively with the Protein Preparation Wizard. The Protein Preparation Wizard panel can be opened from the Workflows menu on the main toolbar. When the preparation has finished, you should check that all changes made are correct. You can fix any remaining errors with the procedures below. For more information on the Protein Preparation Wizard, see the *Protein Preparation Guide*.

Procedures for fixing bond orders, formal charges, and atom names are given below. These procedures use the Build toolbar and the Build panel, which you can display by clicking Show/Hide the Build toolbar on the main toolbar or opening the panel from this button menu.

To assign bond orders:

1. Choose Assign Bond Orders from the Tools menu.
2. Inspect the residues in the structure for any remaining bond orders that are incorrect.
3. If there are still incorrect bond orders, click the Increment bond order or Decrement bond order button on the Build toolbar.

   

4. Click the bonds in the Workspace structure that needs correction.

   You can also right-click the bonds in the Workspace structure, and choose the correct order from the Order submenu of the shortcut menu.

To view and change formal charges:

1. From the Label atoms button menu on the main toolbar, choose Formal Charge.

   Charges are displayed for atoms that are charged.

2. Click the Increment formal charge or Decrement formal charge button on the Build toolbar.

   

3. Click on an incorrectly charged atom in the Workspace until its charge is correct.

   The formal charge is incremented or decremented by one unit for each click.

To change PDB atom names:

1. From the Label atoms button menu on the main toolbar, choose PDB Atom Name.

2. In the Atom Properties tab of the Build panel, choose PDB Atom Name from the Property option menu.
3. Ensure that Pick is selected in the Apply PDB atom name section, and that Atoms is chosen from the Pick option menu.
4. Zoom in on one of the residues (middle+right mouse buttons or mouse wheel).
5. For each atom that needs renaming, enter the desired PDB atom name in the PDB atom name text box, then click on the atom.
6. Repeat Step 4 and Step 5 for each residue whose atoms need renaming.

To change residue names:

1. From the Label atoms button menu on the main toolbar, choose Residue information.

2. In the Residue Properties tab of the Build panel, choose Residue Name from the Property option menu.
3. Ensure that Pick is selected in the Apply residue PDB name section, and that Residues is chosen from the Pick option menu.
4. For each residue that needs renaming, enter the desired PDB name in the Residue PDB name text box, then click on the residue.

   You might need to zoom in to select the residues (middle+right mouse buttons or mouse wheel).

7.2 Using the Refinement Panel

The four tasks that can be performed in the Refinement panel are single-point energy calculation, loop refinement, side-chain prediction, and minimization. These tasks are discussed in the following sections. The general procedure for running a structure refinement is given below.

To run structure refinement tasks:

1. Display the structure you want to refine in the Workspace.

   If the structure has alternate positions, choose the desired set of alternate positions.

2. Choose Applications > Prime > Refinement from the main Maestro window.

   The Refinement panel is displayed.

3. Select a task from the Task menu.

   The default task is Refine loops.

4. If you want to use an implicit membrane model, select Use implicit membrane, then click Set Up Membrane to define the thickness and orientation of the membrane.

5. Select the region for refinement, as appropriate.

   When you have selected a region for refinement, the Start and Write buttons become available.

6. Click Start.

   The Start dialog box is displayed. If you want to run the job later, using `refinestruct`, you can click Write to create the input files.

7. Make changes to the job settings, as appropriate.

8. Click Start.

   The refinement job begins and the Monitor panel is displayed.

The Start and Write buttons remain dimmed until you have specified some part of the structure for refinement. When you have made a selection, the Start and Write buttons become available. The Write button writes the input file for the job, which you can run at a later time

When selecting regions for refinement, you may want to view information about possible structural problems. This information is available in the Protein Reports panel, which can be opened from the Tools menu on the main menu bar. For more information on this panel, see Section 9.5.3 of the *Maestro User Manual*.

The OPLS_2005 force field is used for all refinement tasks.

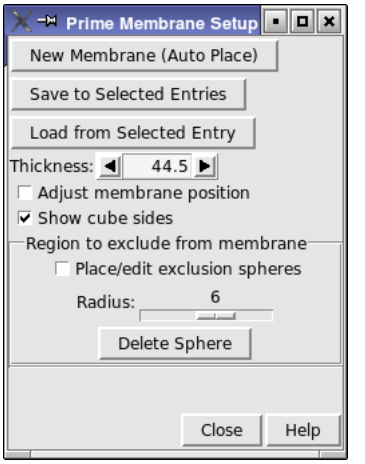

Figure 7.1. The Prime Membrane Setup panel.

7.3 Setting Up an Implicit Membrane

The Prime implicit membrane model is intended for proteins embedded in a membrane. The implicit membrane is a low-dielectric slab-shaped region, which is treated in the same way as the high-dielectric implicit solvent region. Hydrophobic groups, which normally pay a solvation penalty for creating their hydrophobic pocket in the high dielectric region, do not have to pay that penalty while in the membrane slab. Conversely, hydrophilic groups lose any short-ranged solvation energy from the high dielectric region when moving into the low dielectric region.

The implicit membrane model is intended for use with proteins that span the membrane. The slab is trimmed for efficiency, so that any structure that is too small or too far away from the membrane is likely to result in too much trimming and might not produce useful results.

The membrane is marked in the Workspace by a semitransparent cube with a white line that is perpendicular to the surfaces of the membrane (which are planar). The membrane surfaces are colored red, and the other faces of the cube (which are interior to the membrane) are colored green. You can control whether the sides of the cube are displayed with the Show cube sides option. The direction of the line shows the orientation of the membrane.You can adjust both the thickness and the orientation of the membrane.

To set up an implicit membrane:

1. Click Set Up Membrane.

   The Membrane Setup panel opens.

2. Click New Membrane (Auto-Place).

   A membrane is placed, marked in the Workspace as described above.

3. Select Adjust membrane position.

4. Use the middle mouse button to rotate the membrane and the right mouse button to translate the membrane.

5. Adjust the thickness by entering a value in the text box or using the arrow buttons to adjust the thickness up or down by 0.1 Å at a time.

6. Deselect Adjust membrane position.

   This is necessary to exit membrane adjustment and return to normal Workspace operations.

If you want to add the membrane to the entries that are selected in the Project Table, click Save to Selected Entries. The membrane is stored as a set of coordinates for each end of the white line. The entries that you select should therefore contain proteins that occupy approximately the same region of space as the protein used to define the membrane. Otherwise, you will have to adjust the membrane location for these entries.

If you already have a membrane defined for another project entry, you can load it for the current entry by selecting the entry that has the membrane and clicking Load from Selected Entry.

If the protein has pockets that would be occupied by solvent (water), you can select these regions to exclude from the implicit membrane using the tools in the Region to exclude from membrane section. The solvent dielectric constant is then used for these excluded regions.

Excluded regions are defined by a set of spheres that are placed on atoms adjacent to the region. The spheres can overlap, and should cover the entire excluded region. It does not matter that they overlap the protein, because the protein dielectric is not affected. To define a region, select Place exclusion sphere and pick atoms in the Workspace that are adjacent to the region. For each pick a sphere is placed on the atom. You can remove the sphere by picking the atom again. To adjust the sphere volume, use the Exclusion sphere radius slider. This slider only uses integer values, because the exact size of the sphere isn't critical: the spheres only has to cover the excluded region.

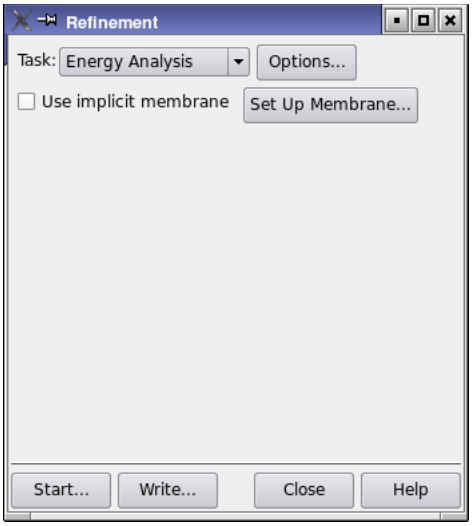

Figure 7.2. The Energy Analysis task in the Refinement panel.

7.4 Prime Energy Analysis

To perform a single-point molecular mechanics energy calculation, choose Energy Analysis from the Task option menu. The calculation uses the OPLS_2005 all-atom force field for protein residues as well as for ligands and cofactors. If desired, set up an implicit membrane model. Click Start, make job settings in the Start dialog box, then click Start to run the job. The output includes a breakdown of the energy into covalent, Coulombic, van der Waals and solvation energy contributions. Each of these is added as a Maestro property to the output structure file, named Prime Covalent, Prime Coulomb, Prime vdW, and Prime Solvation. The output also includes a breakdown of the energy on a per-residue basis.

7.5 Refining Loops

Prime–Refinement is capable of refining loop structures of various lengths, and provides algorithms for different loop lengths. In addition, loops whose structure affects other loops can be cooperatively refined in pairs.

To refine one or more loops serially:

1. Choose Refine Loops from the Task menu.

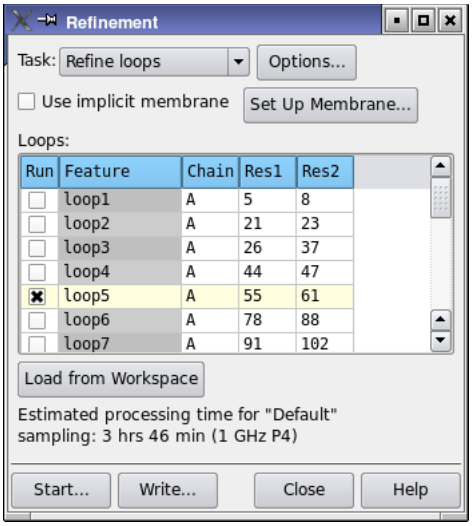

Figure 7.3. The Refine loops task in the Refinement panel.

2. Click Load from Workspace.

   The loop features of the Workspace structure appear in the Loops table. For each loop, the table includes a Run check box, a feature name such as loop1, and the numbers of the first (Res1) and last (Res2) residues in the loop. When you click a row in the table to select a loop, the Workspace shows the loop residues highlighted in yellow. Note that loops of one or two residues are not listed, as they are unsuitable for the refinement protocol.

3. Select a loop by clicking its name in the Feature column.

   The loop row is highlighted in yellow. The default settings for the loop are displayed.

4. If necessary, shorten the loop region to be refined by changing the first and last residue numbers (Res1 and Res2).

   The time required to refine a loop scales approximately linearly with the length of the loop, and the sampling options recommended for loops longer than five residues are also more time-intensive.

5. (Optional) Click Options to select a sampling method and make settings for this loop in the Structure Refinement Options dialog box.

   If the loop is longer than five residues, you should select a sampling method other than the default.

   See Section 7.8.3 for information on the loop refinement options.

6. To designate this loop for refinement, select the check box in the Run column.

   The Start and Write buttons becomes available.

7. Repeat Step 3 through Step 6 for each loop you want to refine.
8. (Optional) Set up and enable an implicit membrane model.
9. Click Start.

   The Start dialog box opens.

10. Make any job settings, then click Start.

    The loop refinement job is started.

When you choose several loops for refinement, they are refined in series. The first selected loop is refined, the best scoring structure for that loop is determined, that structure is used in refining the next loop, and so on. When all loops have been refined, one final structure is returned. If only one loop is refined, the four highest scoring structures are returned by default.

If you want to refine two loops cooperatively, you should replace Step 3 through Step 7 above with the following steps:

1. Click the Run column for the two loops you want to refine cooperatively.
2. If necessary, shorten the loop region to be refined by changing the first and last residue numbers (Res1 and Res2).
3. Click Options to open the Structure Refinement Options dialog box.
4. Select Cooperative loop sampling in the Loop refinement section.
5. Set any other options, then click OK.

For technical details about loop refinement, see Section 7.9.1.

When a loop refinement begins, a validation program checks the loop features of the structure. Apparent errors are reported in a warning dialog box. You may choose Run All Features, Run Only Valid Features, or Cancel. It is recommended that you make a note of the invalid features, click Cancel, and correct the structure if appropriate.

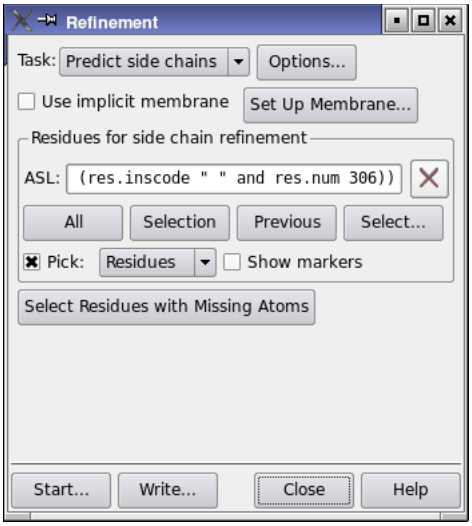

Figure 7.4. The Predict side chains task in the Refinement panel.

7.6 Predicting Side Chains

To predict side-chain conformations:

1. Select Predict side chains from the Task options menu.

   The panel displays atom selection options under the heading Residues for side chain refinement.

2. Use the atom selection options to specify the residues for which you want to predict or refine side-chain conformations (See Chapter 5 of the *Maestro User Manual* for information on selecting atoms).

   If you imported a PDB structure that has residues with missing atoms, you can select those residues by clicking Select Residues with Missing Atoms. The selection is based on the information generated when the structure was imported, not on the current structure. The selection therefore includes any residues that were fixed since import.

3. Click Options to change the Number of structures to return from the default, which is to return a single side-chain prediction.

4. Click the Run button to start the side-chain prediction job.

You can also use this process to add side chains to a backbone structure from Refine Backbone.

For technical details about side-chain prediction, see Section 7.9.3.

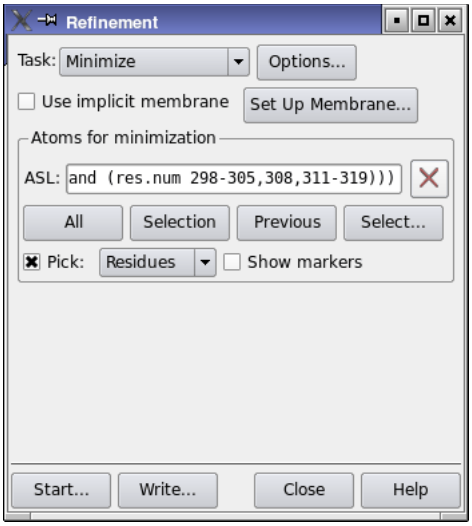

Figure 7.5. The Minimize task in the Refinement panel.

7.7 Minimizing Structures

The Minimize refinement task performs a truncated-Newton energy minimization, using the OPLS_2005 all-atom force field for proteins as well as for ligands and cofactors, and treating solvation energies and effects via the Surface Generalized Born (SGB) continuum solvation model.

To minimize all or part of the structure selected:

1. Select Minimize from the Task menu.

   Atom selection options are displayed under the heading Atoms for minimization.

2. Click All to minimize all residues, or specify a set of atoms or residues.

   For more information on atom selection, see Chapter 5 of the *Maestro User Manual*.

3. Click Start to start the minimization job.

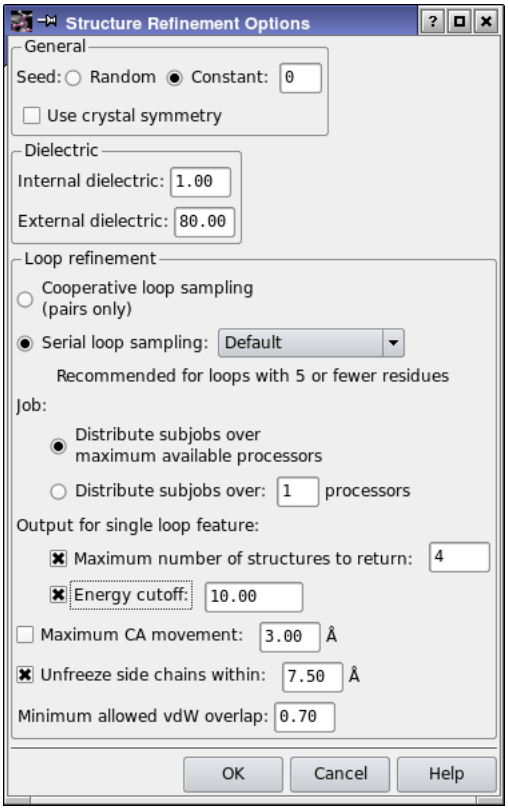

Figure 7.6. The Structure Refinement Options dialog box.

7.8 Refinement Panel Options

The Structure Refinement Options dialog box offers three sets of options: General options, Dielectric options, and options for Loop Refinement. Click Options in the Refinement panel to view or change the options and settings.

7.8.1 General Options

The General section supplies options that apply to all types of refinement. The options include specification of a random-number seed for starting the side-chain prediction and use of crystal symmetry.

Seed

Specify the seed for the random-number generator used in side-chain predictions. The seed is used for side-chain prediction in both the loop refinement and the side-chain prediction tasks. The options are:

- Random: use a random seed.
- Constant: use the integer given in the text box as the seed. The default is 0.

Use crystal symmetry

If crystal symmetry is known for this protein, apply periodic boundary conditions so that the crystal symmetry is satisfied.

7.8.2 Dielectric Options

The Dielectric section allows you to set the dielectric constants used in the calculations, in the following text boxes.

Internal dielectric

Set the dielectric constant used for the interior of the protein and any implicit membrane.

External dielectric

Set the dielectric constant for the (continuum) solvent.

7.8.3 Loop Refinement Options

The Loop Refinement section provides options for selecting the sampling method, job options, output options, and some general options.

Prime can perform cooperative loop sampling as well as serial loop sampling. Both the size of the loop and the sampling options chosen affect the time needed for loop refinement. The options are:

Cooperative loop sampling

Select this option to refine two loops together. Each loop is refined in turn until the refinement has converged for both loops. You must have only two loops selected to use this option.

Serial loop sampling

Select this option to refine each loop independently. The option menu provides five levels of sampling accuracy: Default, Extended Low, Extended Medium, Extended High, Ultra Extended.

The text below the option and menu displays the recommended loop length for the selected accuracy.

Extended sampling samples loop conformations more thoroughly, using a series of loop constraint settings. Up to eight processors can be used simultaneously to perform loop refinement. Extended sampling can be run at four different sampling levels: Low, Medium, High, and Ultra. If you select Ultra Extended, you can only refine one loop at a time.

Job

These options allow you to set a limit on the number of processors over which to distribute the loop sampling subjobs. Each sampling method runs in several stages, and each stage can use a different number of processors. The maximum number of processors that can be used by the different sampling methods are:

Default	1
Extended Low	2
Extended Medium	4
Extended High	8
Ultra Extended	8*L (L is the loop length)

- To make use of as many processors as are available, select Distribute subjobs over maximum available processors.
- To set the maximum number of processors, select Distribute subjobs over and enter a value in the text box.

Output for single loop feature

These options determine how many structures are returned:

- Maximum number of structures to return: Select this option to return more than one structure for each loop specified for refinement. The default when this option is selected is to return 4 structures for each loop.
- Energy cutoff: Select this option to prevent higher-energy structures from being returned. Structures whose energy is higher than that of the lowest-energy structure by more than this amount are not returned. The default cutoff when this option is selected is 10 kcal/mol.

The following settings are applied to the loops you have selected for refinement.

- Maximum CA movement: To limit the extent to which alpha carbon atoms can move from their original positions, select this option, and enter a limit in the text box. The default distance is 3.0 Å. The extended sampling procedure tests multiple values for Maximum CA movement, so this option is unavailable when extended sampling is selected.

- Unfreeze side chains within: Select this option to allow side chains near the specified loop to move, and specify a distance in the text box. The default distance is 7.5 Å.
- Minimum allowed vdW overlap: By default, this value is set to 0.70, meaning that the distance between atoms must be at least 70% of their 'ideal' van der Waals separation. The extended sampling procedure tests multiple values for the minimum overlap, and therefore this option is unavailable when extended sampling has been selected.

7.9 Technical Details for Refinement

7.9.1 Loop Refinement

Loop refinement using the Default option proceeds as follows:

1. The loop is reconstructed using the backbone dihedral library, by building up half from each direction. The resolution starts off very coarsely, becoming finer until it manages to produce a required number of physically realistic loop conformations (based on the length of the loop).
2. The large number of loops generated in the first step are clustered, and representatives of each cluster are selected.
3. These cluster representatives are then scored as follows:
    a. Side chains are re-added to the loop residues, as well as any residues within a specified cutoff, according to the side-chain optimization procedure.
    b. The loop and contact side chains are minimized.
    c. The energy is calculated.

4. The best scoring loop structures are returned.

Loop prediction with extended sampling is specifically designed to overcome sampling problems with long loops, in which the number of residues results in an unwieldy number of possible loop conformations.

Extended sampling uses the following procedure:

1. Two initial predictions are run on the loop. These are intended to coarsely sample conformational space and are carried out with two slightly different sets of prediction parameters. Each initial prediction returns up to four top-scoring conformations, resulting in up to eight structures. These eight structures are intended as initial points for finer sampling of what appears to be favorable regions (basins) of conformational space.
2. Each of the eight structures generated in Step 1 is run through another stage of refine-

ment, with a 4 Å restriction on Cα movement. This is intended to sample the basins more finely.

3. The eight refined structures from Step 2 are combined with the original structures from Step 1, and this set of 16 structures is ranked by energy. The five top-scoring structures are then subjected to a final stage of refinement, with a 2 Å restriction on Cα movement. This is intended to perform fine-grained sampling of the best basins.

This procedure defines the "Extended High" sampling procedure. The "Extended Medium" and "Extended Low" carry four and two structures, respectively, into Step 2 and Step 3.

In ultra-extended sampling, the three steps of extended sampling are followed by the steps below.

4. The top four structures are selected, and two predictions run on each, with part of the loop frozen. Runs are done with "moving windows" with varying numbers of contiguous residues: all *n* residues, the two possible contiguous sets of *n*-1 residues, the three sets of *n*-2 residues, and so on. The structures are sorted after each cycle of a given number of residues.

5. Step 2 is repeated with the top eight structures cumulatively generated, then Step 3 is repeated with the top eight structures cumulatively generated.

7.9.2 Cooperative Loop Refinement

Cooperative loop refinement proceeds as follows:

1. A list of side chains that will be optimized is generated. By default this list includes any residue with an atom within 7.5 Å of either loop 1 or loop 2.

2. Loop 1 is sampled, allowing side chains in the list from Step 1 to move, holding the backbone of loop 2 fixed at its original conformation. Eight loop conformations are generated.

3. Loop 2 is sampled, allowing side chains in the list from Step 1 to move, holding the backbone of loop 1 fixed at its original conformation. Eight loop conformations are generated. This step runs concurrently with Step 2.

4. Conformations of loop 1 from Step 2 with loop 2 from Step 3 are merged into a set of structures.

5. All side chains in the list determined in Step 1 are reoptimized, keeping only those structures whose energy is below the energy cutoff.

6. All residues in both loops are then minimized, including the backbone, keeping only those structures whose energy is below the energy cutoff.

7. The 4 best new structures are taken as input for resampling, starting from Step 2. The process is repeated until convergence is obtained.

7.9.3 Side-Chain Prediction

The structure refinement program uses the following procedure to re-predict conformations for the side chain set that you select.

1. Side-chain rotamers are randomized for nonconserved residues (the default) or for all residues.
2. Beginning with the first residue to be predicted, the side-chain rotamer library is used to find the rotamer with the lowest energy while keeping all other side chains fixed. Once the process is complete for the first residue, the next residue is treated, and so forth until all have been done once (a single pass has been completed).
3. Once the pass is complete, Step 2 is repeated from the beginning, and then repeated again until the side-chain rotamers appear to be converged (no more changes are occurring).
4. Minimization is run on all of the side-chain atoms (but not backbone atoms) of the residues being treated.

7.10 Refinement with Nonstandard Residues

If you want to refine a structure with nonstandard residues, you can make changes to the residues in the structures before refinement. Several nonstandard residues are supported in the Build panel: the neutralized forms of ASP, LYS, ARG, and GLU, which are named ASH, LYN, ARN, and GLH; and the tautomers of HIS (HID=HIS and HIE) and its protonated form, HIP.

Ionization (deprotonation) of four other residues are supported, CYS, SER, THR, and TYR. The names of the ionized residues are CYT (thiolate), SRO (alkoxide), THO (alkoxide), and TYO (phenoxide). In addition, the disulfide-bridged CYS is supported as CYX. To change one of these residues, you can either change the name, or change the structure. For example, you can generate an ionized SER by either of the following methods:

- Changing the residue name to SRO. The HG is deleted and a formal negative charge is added to the OG.
- Deleting the HG (if present) and adding a formal charge to the OG. The residue is renamed to SRO.

7.11 Output of Refinement Jobs

By default, the structures created in Prime–Refinement are appended to the Project Table for the project currently open. If you would prefer the output structure to replace the input structure, or would rather not incorporate the output structure into the project at all, select the appropriate option from the Incorporate option menu in the Start dialog box:

- Append new entries as a new group
- Append new entries individually
- Replace existing entries
- Do not incorporate

In addition to returning the structures, the output of a refinement job include several Maestro properties. The main properties are Prime Energy and Prime Energy Difference, which reports the difference in energy between the input and output structures. The energy difference is cumulative: if the property already exists, the difference is incremented in the current job. For loop refinement and side-chain prediction jobs, the lipophilic term is reported separately as Prime Lipo; the sum of the Prime energy and the lipophilic term is used to rank loop conformations.

Maestro Protein Structure Alignment

It is sometimes useful to be able to align proteins outside the usual Prime workflow. Maestro provides access to the structure alignment facilities of Prime so that you can perform such alignments. Two panels are available from the tools menu: the Protein Structure Alignment panel, and the Align Binding Sites panel.

8.1 The Protein Structure Alignment Panel

To open the Protein Structure Alignment panel, choose Protein Structure Alignment from the Tools menu in the main window.

In the Protein Structure Alignment panel, structure alignment is performed on Project Table entries that have been included in the Workspace. The reference structure, whose frame of reference is used for the alignment, must be the first included entry (the entry with the lowest row number). To make another protein the reference, move that protein above all the other included entries in the Project Table.

By default, the alignment includes all residues. However, it is often useful to align a subset of residues that have greater similarity than the structures as a whole, or are more informative to compare. Because this program uses matching of secondary structure elements, the subset should have enough contiguous residues to include at least one helix or strand. Very small numbers of residues do not produce useful alignments. Selection of subsets is discussed further under Residues to align, below.

When the alignment calculation is complete, the structures are placed in the same frame of reference in the Workspace, and the results of the calculation are listed in the text display area of the Protein Structure Alignment panel. The aligned residues are listed and an RMSD and Alignment Score are reported. The RMSD is calculated based only on those residues that are considered to have been successfully aligned, and therefore does not represent an overall comparison of the structures. It is computed from the C-alpha atoms of the aligned residues. An Alignment Score lower than 0.6–0.7 indicates a good alignment. Alignment Scores greater than about 0.7–0.8, or a failure of the structural alignment calculation, indicates there is insufficient structural similarity for a meaningful alignment. For details on the algorithm, see the paper by Honig and Yang (*J. Mol. Biol.* **2000**, *301*, 665).

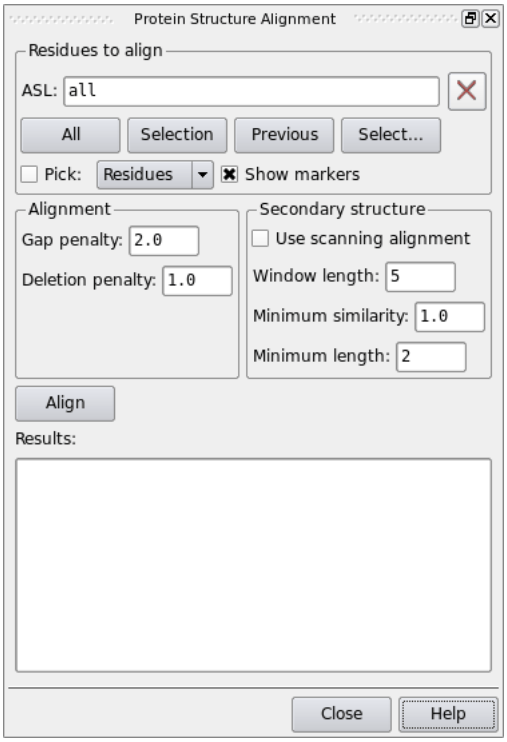

Figure 8.1. The Protein Structure Alignment panel.

To run a default alignment job, include two or more protein structures in the Workspace, open the Protein Structure Alignment panel, and click Align to start the alignment calculation.

The Protein Structure Alignment panel provides some tools for controlling which parts of the protein are aligned and how they are aligned. These tools are described below.

Residues to align Section

Alignment can be performed for all residues or for a subset of residues. You might need to try different subsets to obtain the most useful alignment. You can use the picking controls, Workspace selection, or the Atom Selection dialog box to specify subsets. Subsets should include at least one secondary structure element (helix or strand). Clicking the Select button opens the Atom Selection dialog box to the Residue tab. Here you can select a range of residue numbers, specify portions of the residue sequence, or click Secondary Structure in the list of options on the left to choose residues based on their location in Helix, Strand, or Loop structures. Subsets can be combined, intersected, or otherwise modified in the Atom Selection dialog box.

Alignment Section

Gap penalty

In the structure alignment calculation, this is the alignment score penalty for initiating a gap in the alignment. The default gap penalty is 2.00. The main purpose of this parameter is to prevent the generation of very poor initial alignments.

Deletion penalty

In the structure alignment calculation, this is the penalty for extending a gap. The default deletion penalty is 1.00.

Secondary Structure Section

Use scanning alignment

This option is deselected by default. When it is selected, structure alignment is carried out in scanning rather than the default global mode. Generally, the default global mode is more useful. However, if you do not find any similarity between structures with global alignment, you can try scanning alignment.

Global alignment involves running double dynamic programming on the full scoring matrix built from all secondary structural elements (SSEs) in both structures. In scanning alignment double dynamic programming is run on successively smaller subregions of the full scoring matrix (window sizes of 20, 15, 10 and 5 SSEs) until a reasonable alignment is produced. Assuming a reasonable alignment is found it is used as a starting point for building a global alignment (involving all SSEs).

Window length

Number of successive SSEs that are to be used as a baseline for the global alignment. The default is 5. This option is significant only if Use scanning alignment has been selected.

Minimum similarity

This is the alignment score above which alignments are reported as unsuccessful. The default is 1.00, where alignment scores of 0.7 or less mean greater similarity, and alignment scores above 1.0 mean less similarity. When alignment scores are greater than 1.0, the alignment is unlikely to be meaningful.

Structural similarity of proteins is measured by the Protein Structural Distance (PSD). This measure combines the similarity scores for the secondary structural elements obtained from dynamic programming and the RMSD of aligned residues for the optimal alignment. For details see Yang, A.; Honig, B. *Journal of Molecular Biology* **2000**, *301*, 665-678.

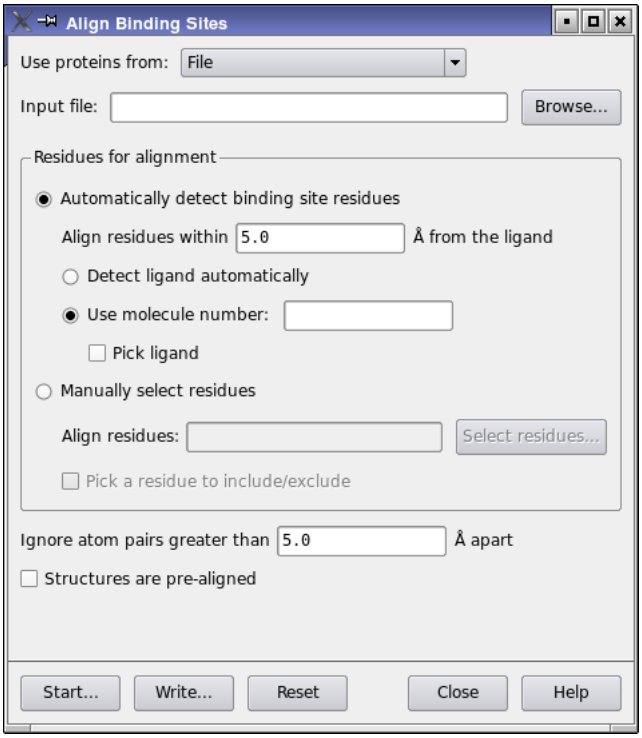

Figure 8.2. The Align Binding Sites panel.

Minimum length

Minimum number of SSEs that are to be aligned in scanning alignment. The default is 2.

Results Text Area

Results from the alignment job are displayed in this text area.

8.2 The Align Binding Sites Panel

The Align Binding Sites panel allows you to align the binding sites (or other structural features) of members of a family of proteins. The Align Binding Sites job first runs a Protein Structure Alignment to obtain the global alignment and then automatically generates the list of atoms to use in a pairwise alignment of the C-alpha atoms from the residues selected.

To open the Align Binding Sites panel, choose Tools > Align Binding Sites in the main window.

The Use proteins from option menu provides choices of the source of protein structures. If you choose Project Table, the entries that are selected in the Project Table are used. If you choose File, the Input file text box and Browse button become available, and you can browse for the input file.

In the Residues for alignment section you select the residues that are to be used to align the proteins. There are two options, described below.

- Automatically detect binding site residues—Align residues within a specified distance of the ligand. The distance is specified in the Align residues within *N* Å from the ligand text box. Two options are available for specifying the ligand:
  - Detect ligand automatically—Detect the ligand by selecting the molecule that is most ligand-like (number of atoms).
  - Use molecule number—Specify the ligand by molecule number. You can enter a number in the text box, or select Pick ligand and pick a ligand atom in the Workspace.
- Manually select residues—Specify the residues to align by using any of the following methods:
  - Enter the residue specification in the Align residues text box, in the format *chain*:*number*.
  - Click Select residues to open the Atom Selection dialog box and define the residues.
  - Select Pick a residue to include/exclude and pick residues in the Workspace. Picking an already included residue excludes it.

  The residues are listed in the Align residues text box, which you can edit to remove or add residues.

The alignment is performed on atom pairs. You can ignore atom pairs in the alignment process if they are further apart than a specified distance, by entering the distance in the Ignore pairs greater than *N* Å apart text box.

If you have proteins that are prealigned, you can select Structures are prealigned to suppress the global alignment of the protein structures, and align only by using the selected residues.

When you have finished making settings, click Start to start the job.

You can also run a job to align binding sites from the command line—see Section 12.4 on page 119 for more information.

Docking Covalently Bound Ligands

While Glide is highly efficient at docking ligands that are bound to a receptor through hydrogen bonds or various nonbonded interactions, it is not able to dock ligands that are covalently bound to the receptor. Prime provides a covalent docking facility that allows you to select the attachment point to the receptor and the possible attachment points on the ligand, and run a Prime loop prediction to find poses for the ligand.

9.1 Running Covalent Docking from Maestro

Covalent docking calculations can be set up and run from the Covalent Docking panel. To open the Covalent Docking panel, choose Applications > Prime > Covalent Docking in the main window.

The Covalent Docking panel has two main sections. Above these sections, you can specify the source of the ligands. In the Ligand reactive groups section, you can specify the kinds of groups that will react at the receptor attachment site. In the Receptor options section, you can specify the receptor bond that is broken and choose receptor residues to sample.

9.1.1 Selecting the Ligands

The ligands that you dock to the receptor can be taken from the Project Table or from a file.

- To use the structures that are selected in the Project Table, choose Project Table from the Use ligands from option menu.
- To read the ligands from a file, choose File from the Use ligands from option menu. You can then either enter the file name in the Input file text box, or click Browse and navigate to the file in the file selector that opens. The file must be a Maestro file, and can be compressed (`.maegz`, `.mae.gz`) or uncompressed (`.mae`).

You should ensure that the structures are all-atom, 3D structures that are properly prepared, for example by using LigPrep. See the *LigPrep User Manual* for more information on ligand preparation.

The poses that are returned include both the ligand and the receptor, so you should consider carefully how many ligands you want to dock.

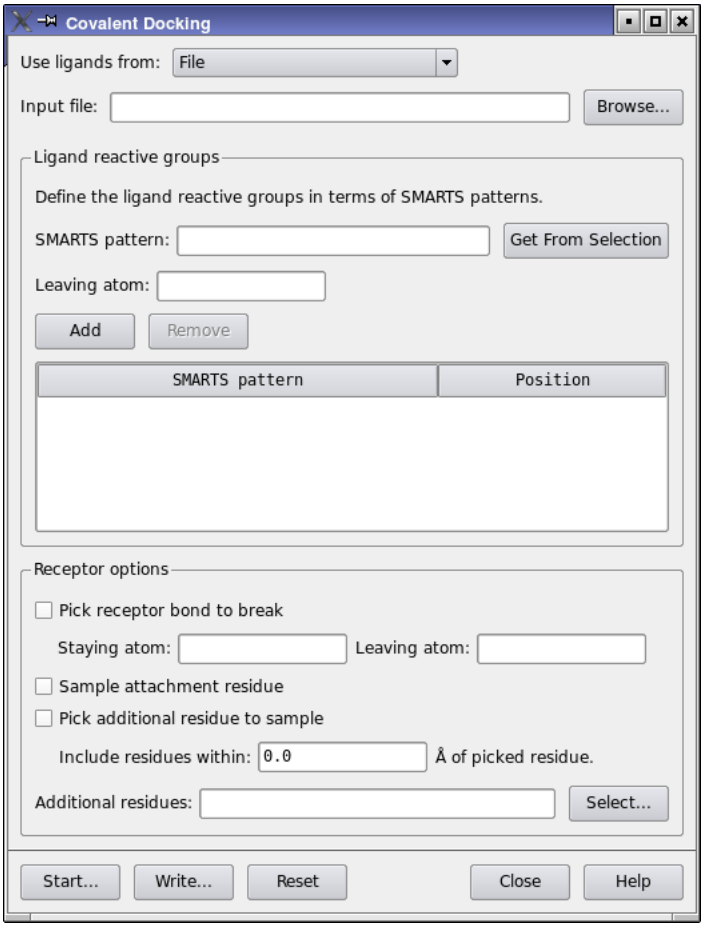

Figure 9.1. The Covalent Docking panel.

9.1.2 Defining the Ligand Reactive Groups

In the Ligand reactive groups section, you specify the reactive groups on the ligands, in which a bond is broken to form a bond with the receptor. Each group is specified by a SMARTS pattern and the index of the atom that leaves, with its attachments, when the ligand bond is broken. You can specify multiple reactive groups. For each ligand, all groups are tried for a match to the ligand, and each match is docked.

To define each SMARTS pattern, you can enter the pattern in the SMARTS pattern text box, or you can select atoms on a typical ligand molecule in the Workspace, and click Get From Selection to place a pattern in the SMARTS pattern text box. You can then edit the SMARTS pattern in the text box if you want.

When you are satisfied with the SMARTS pattern, you must then designate an atom on the ligand that is removed, along with the atoms attached to it, when the ligand forms a bond with the receptor. The fragment with the smaller number of atoms is the part that is removed. The atom is specified by its index (position) in the SMARTS pattern; the first atom index is 1. This index must be entered into the Leaving atom text box.

Having defined a SMARTS pattern and the leaving group, click Add. The SMARTS pattern is added to the table below, and will be used when processing the ligands. You can repeat this process for as many SMARTS patterns as you want to use. If you want to remove a pattern from the table, select it and click Remove.

9.1.3 Specifying the Receptor Bond to Break

In the Receptor options section you specify the bond in the receptor that is broken to form a bond with the ligand. The receptor must be displayed in the Workspace, and is written to a file when you start the job.

To define the bond that is broken, select Pick receptor bond to break, and pick the desired bond on the receptor in the Workspace. The Staying atom and Leaving atom text boxes are filled with the atom numbers of the atom in the bond that remains and the atom in the bond that is removed when the bond is broken. The bond is marked in the Workspace with yellow markers. If you pick the wrong bond, you can simply pick again, and the new bond replaces the old.

You can also enter or edit the atom numbers in the Staying atom and Leaving atom text boxes. To ensure that you have the correct atom numbers, you should label the atoms. For more information on labeling atoms, see Section 6.3 of the *Maestro User Manual*.

9.1.4 Specifying Receptor Residues to Sample

In the Receptor options section you can also choose residues that are allowed to adjust to the formation of the covalent complex in the loop sampling procedure.

The first choice is whether to sample the residue that is attached to the ligand. To do so, select Sample attachment residue.

The next step is to pick other residues in the Workspace for sampling. To do so, select Pick additional residue to sample, and pick residues in the Workspace. You can pick any number of residues. As well as picking, you can select the desired residues in the Atom Selection dialog box, which you open by clicking Select. The residues that you select by either method are listed in the Additional residues text box.

You can extend your selection to include residues that are within a given distance of the selected residues. To do so, enter a value in the Include residues within N Å of picked residues text box. These residues are also added to the list in the Additional residues text box.

You can remove residues from the list to sample by deleting them from the Additional residues text box.

9.1.5 Running the Job

When you have finished making settings, click Start to open the Start dialog box. In this dialog box you can make job settings and start the job.

The results are returned in a Maestro file. Each pose includes both the ligand and the receptor, one pose for each ligand attachment point found.

9.2 Running Covalent Docking from the Command Line

It is also possible to run covalent docking jobs from the command line. To do so, you should first set up the input file in the Covalent Docking panel, as described in the previous section, and then click Write, to write the input file for the job. This button opens a file selector, in which you can browse to the location and choose the file name.

The covalent docking job is run with the same pipelining workflow scripts as other workflows, such as the Virtual Screening Workflow (VSW) and Quantum-Polarized Ligand Docking (QPLD). Covalent docking does not have its own script, so you can use the script for VSW, and run the job as

`$SCHRODINGER/vsw [`*job-options*`] `*input-file*

The job options are the usual Job Control options that specify the host and other resources. These options are described in Section 2.3 of the *Job Control Guide*.

Prime MM-GBSA

The Prime MM-GBSA panel can be used to calculate ligand binding energies and ligand strain energies for a set of ligands and a single receptor, using the MM-GBSA technology available with Prime.

The ligands and the receptor must be properly prepared beforehand, for example, by using LigPrep and the Protein Preparation Wizard. The ligands must be pre-positioned with respect to the receptor, and the receptor must be prepared as for a Prime refinement calculation.

To open the Prime MM-GBSA panel, choose MM-GBSA from the Prime submenu of the Applications menu.

After preparing your structures, you can set up the calculation as follows:

1. Specify the source of the structures.

   You can take structures from a Pose Viewer file (Glide output), or from separated ligand and protein structures. If you choose the latter option, you must ensure that the ligands and the protein are properly prepared and aligned.

2. (optional) Choose calculation settings.

   If you want to evaluate ligand strain energies as well as ligand binding energies, select Calculate ligand strain energies. With this option, an extra calculation on the ligand is performed at its geometry in the complex, and this result is combined with the free ligand calculation to calculate the strain energy.

   If you want to use ligand partial charges evaluated by some other program, such as QSite, select Use ligand input partial charges. The input partial charges on the ligand will then be used instead of those assigned using the default force field.

   If you want to use the Prime implicit membrane model, select Use implicit membrane. The receptor you choose must have been already set up with the membrane, as described in Section 7.3 on page 56.

3. (optional) Select a region within a certain distance of the ligand for which the protein structure will be relaxed in the calculation.

   The atoms in all residues within the specified distance of the first ligand processed are included in the flexible region. By default, all protein atoms are frozen, and only the ligand structure is relaxed. The larger the flexible region, the longer the calculation takes.

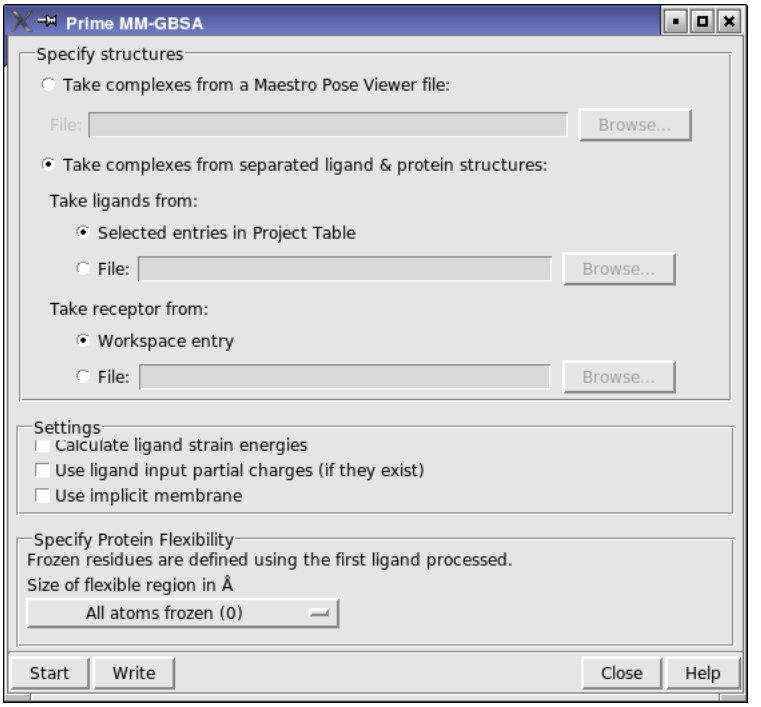

Figure 10.1. The Prime MM-GBSA panel.

4. Click Start to start the job, or click Write to write the input file and run the job from the command line.

   When you click Start, the Start dialog box opens, in which you can choose to incorporate the results into the Maestro project, set the job name, select the host, and set the number of CPUS and number of subjobs, if you are running the job on a multiprocessor host or submitting it to a queuing system. You can divide the ligand set between a specified number of subjobs, to achieve load balancing. The number of subjobs must be no smaller than the number of CPUs, and for optimal load balancing, should be several times the number of CPUs.

You can also run Prime MM-GBSA from the command line on Unix, with the input file written from the panel. The syntax is as follows:

`$SCHRODINGER/prime_mmgbsa [`*options*`]` *input-file*

The input file must be a Maestro file with the receptor as the first entry followed by one or more ligands (i.e. pose viewer file format). The options are described in Table 10.1.

Table 10.1. Options for the prime_mmgbsa command

Option	Description
`-b` *jobname*	Use alternative job basename. Default is to use the input file basename.
`-cmae`	Use atomic partial charges from the input Maestro file in simulations
`-extdiel` *value*	Set the external dielectric to value.
`-h`	Print help message and exit
`-intdiel` *value*	Set the internal dielectric (including the membrane) to *value*.
`-keep`	Save all minimized ligand, complex, and protein structures (default is to save only minimized protein-ligand complexes)
`-lstrain`	Include an estimate of the ligand strain energy in the output
`-membrane`	Use the Prime implicit membrane model in protein and complex simulations
`-n` *subjobs*	Number of subjobs to run.
`-rflex` *file*	File listing protein residues to be treated flexibly. The residues are specified in the format *chain*:*residue*.
`-v`	Print version information and exit

The output Maestro structure file includes a number of properties, which are described in Table 10.2.

Table 10.2. Output properties from a Prime MM-GBSA calculation.

Property	Description
Prime Coulomb	Coulomb energy of the complex
Prime Covalent	Covalent binding energy of the complex
Prime vdW	Van der Waals energy of the complex
Prime Solv SA	Solvation surface area of the complex
Prime Solv GB	Solvation energy of the complex
Prime Energy	Prime energy of the complex
Prime MMGBSA Complex Energy	MMGBSA energy of the complex
Prime MMGBSA Ligand Energy	MMGBSA energy of the free ligand
Prime MMGBSA Receptor Energy	MMGBSA energy of the uncomplexed receptor

Table 10.2. Output properties from a Prime MM-GBSA calculation. (Continued)

Property	Description
Prime MMGBSA Ligand in Complex Energy	MMGBSA energy of the ligand in the complex
Prime MMGBSA Ligand Strain Energy	MMGBSA ligand strain energy
Prime MMGBSA DG bind	MMGBSA free energy of binding
Prime MMGBSA DG bind no Ligand Strain	MMGBSA free energy of binding without including ligand strain

Command Syntax

This chapter summarizes the command syntax for various programs that are part of Prime. All the programs are executed by driver scripts, which run under the Schrödinger Job Control facility. For more information on this facility, see the *Job Control Guide*. The conventions used in describing the command syntax are outlined in the Document Conventions section at the front of this manual.

Residue specifications are used with a number of the input file keywords for several of the scripts. These have the format *C*:*N*, where *C* is a one-character chain ID ('_' if the chain ID is blank) and *N* is the residue number and insertion code (if any). Atom specifications similarly have the format *C*:*N*:*A*, where *A* is the 4-character PDB atom name with spaces replaced by underscores, e.g. `A:185:_SG_`.

Most of these scripts accept the standard Schrödinger Job Control command line options and other commonly used options, listed in Table 11.1. Exceptions are noted in the program descriptions.

Table 11.1. Standard Schrödinger job control options.

Option	Description
`-HOST` *host*	run the job on the specified host. The default is the local host.
`-LOCAL`	Run the job in the local directory rather than the temporary directory
`-TMPDIR` *tmpdir*	Run the job in the temporary directory *tmpdir*
`-WAIT`	Do no return control to the shell until the job finishes
`-INTERVAL` *n*	Update the output every *n* seconds
`-NICE`	Run the job at reduced priority
`-DEBUG`	Turn on debugging for job control as well as the program
`-HELP`	Show information on command-line usage

11.1 blast—Run Blast Searches

The `blast` script provides an interface to the Blast and PSIBlast programs from NCBI, while also supporting standard Schrodinger Job Control options. All options are passed on to the script via an input file.

Syntax

```
$SCHRODINGER/blast jobname [options]
```

Input is read from the file *jobname*`.inp`. Output is written by default to *jobname*`.out`.

Options

The `blast` script accepts the options listed in Table 11.1.

Input File

Keywords control the operation of the `blast` script, and are listed in Table 11.2. Most keywords are simply a translation of options available from the `blastall` or `blastpgp` executables distributed by NCBI. For optional keywords, the Blast default values are used. Since these defaults can differ between different version of Blast they are not listed here.

Table 11.2. Keywords for the blast script

Keyword	Description
`QUERY_FILE` *filename*	Required. Name of the FASTA file containing the input/query sequence
`DATABASE` {`nr`|`pdb`}	Database to be used in the search. In a default installation the two allowed values are `nr` and `pdb`.
`ALIGNMENT_FORMAT` *value*	Alignment view options (Blast `-m` option). Allowed values are: `0` pairwise `1` query-anchored showing identities `2` query-anchored no identities `3` flat query-anchored, show identities `4` flat query-anchored, no identities `5` query-anchored no identities and blunt ends `6` flat query-anchored, no identities and blunt ends `7` XML Blast output `8` tabular `9` tabular with comment lines `m2io` Maestro format (default)
`EXPECT` *value*	Expectation value (E) (Blast `-e` option).
`FILTER` {`true`|`false`}	Filter query sequence: DUST with `blastn`, SEG with others. (Blast `-F` option).
`GAP_OPEN_COST` *value*	Cost to open a gap (Blast `-G` option)
`GAP_EXTEND_COST` *value*	Cost to extend a gap (Blast `-E` option)

Table 11.2. Keywords for the blast script (Continued)

Keyword	Description
MATRIX *type*	Sequence similarity matrix to use for the alignments. Allowed values of *type* are BLOSUM45, BLOSUM62, BLOSUM80, PAM30, PAM70
MAX_ROUNDS *n*	Maximum number of passes to use in multipass version (PSI-Blast -j option); only applies to runs using PROGRAM blastpgp
E_VALUE_THRESHOLD *value*	e-value threshold for inclusion in multipass model (PSIBlast -h option).
OUTPUT_FILE *filename*	Name of the output file. Default: *jobname*.out.
PROGNAME *name*	Type of blast search to run. The allowed values are blastp, blastpgp, and bl2seq (for pairwise alignments). Default: blastp.
TEMPLATE_FILE *filename*	Name of the FASTA file that contains the template sequence (bl2seq -j option). Only applies to pairwise alignments using bl2seq
PSSM_ASCII_OUTFILE *filename*	ASCII output file for PSI-BLAST matrix (PSIBlast -Q option).
EXPAND_HITS {true|false}	Expand redundant hits, based on information in the NCBI define for a particular hit. Only available when using Maestro format.
SHOW {all|pdb}	Allows the user to restrict the types of hits that are included in the output. all includes all hits; pdb only includes hits from the PDB. Only available when using Maestro format.

Return Value and Errors

blast returns 0 if successful and a non-zero value in all other cases.

Examples

The following is an example input file; query.aa is the query sequence in FASTA format.

```
QUERY_FILE        query.aa
PROGNAME          blastp
DATABASE          pdb
```

Environment Variables

The following environment variables are supported by the blast script:

PSP_BLASTDB	directory containing the customized Blast databases
PSP_BLAST_DIR	directory containing a custom Blast installation

11.2 bldstruct—Build a Protein Model from an Alignment

The `bldstruct` script can be used to build initial (unrefined) protein models from an alignment to one or more templates via the Protein Local Optimization Program (PLOP). `bldstruct` handles all the tasks associated with the building of homology models via PLOP (i.e. parsing and preparation of input files, interaction with PLOP, and handling of output structures, etc.). All jobs handled by `bldstruct` are run under Schrödinger Job Control.

Multithreaded execution (with Open MP) can be enabled for `bldstruct` jobs by setting the environment variable `OMP_NUM_THREADS`. It is recommended to set this environment variable in the `schrodinger.hosts` file for hosts on which multithreading is supported—see Section 6.1 of the *Installation Guide*.

Syntax

`$SCHRODINGER/bldstruct` *input-file* `[`*input-file2 ...*`]` `[`*options*`]`

The input file can be specified with or without the `.inp` extension. If multiple input files are specified, the chains are combined to build a multimer. Each input file should correspond to a single chain of the multimer, and be accompanied by the usual associated PDB template structure and alignment files. The output is a single structure that contains all of the chains, refined together; the job and the output file are named after the first input file.

Options

The `-DEBUG` and `-HOST` options are recognized by `bldstruct`. See Section 2.3 of the *Job Control Guide* for more information.

Command Input File

The command input file is named *jobname*`.inp`. This input file contains keyword-value pairs, one pair per line separated by one or more spaces. The keywords are given in Table 11.3.

Table 11.3. Keywords for the bldstruct script

Keyword syntax	Description
`BUILD_DELETIONS {true|false}`	Turn on or off closure of the chain breaks near deletions in the alignment. If not closed, they will remain as chain breaks in the output structure. Default: `true`.
`BUILD_TRANSITIONS {true|false}`	Turn on or off closure of the junctions between templates in multi-template homology modelling jobs. If not closed, they will remain as chain breaks in the output structure. Default: `true`.

Table 11.3. Keywords for the bldstruct script (Continued)

Keyword syntax	Description
COMPOSITE_ARRAY *string*	A string of integers indicating from which template (or, more specifically, from which alignment) coordinates for a given residue should be obtained. Thus, this string of integers must be equal in length to the query sequence being used in the alignment files. COMPOSITE_ARRAY is required for multiple template jobs. It is not required for single template jobs, but if it is present, it is ignored.
KEEP_ROTAMERS {true|false}	Specifies which side chains to optimize. If true, the rotamers for conserved side chains are preserved, and only non-conserved side chains are predicted. If false, all side chains are predicted. This keyword is ignored if SIDE_OPT is false. Default: true.
MAX_INSERTION_SIZE *n*	Specifies the longest insertion in the alignment that will be built. Insertions longer than this will be omitted, and not appear in the output structure. Residue numbering will reflect the full query sequence however. Default: 1000.
MINIMIZE {true|false}	Turn on or off minimization of regions that were not directly copied from the template structure. These regions primarily include portions of the structure involved in closing gaps or building insertions, but also include any side chains optimized due to the SIDE_OPT keyword. Default: true.
SIDE_OPT {true|false}	Turn on or off a side chain optimization stage. Default: true.
*template*_ALIGN_FILE *filename*	Required. The name of the file containing the alignment between this template and the query. Details on the format are provided in the Examples section below.
*template*_HETERO_*i* *ligand-spec*	Required if a ligand is present. Specifies a ligand to include from this template. The format of the specification is AAA C:### where AAA is the ligand's three letter code, C is its chain ID, ### is its residue number. Multiple ligands from the template are numbered using *i* as an index, starting from zero.
TEMPLATE_NAME *template*	Required. A label used to identify a given template in subsequent data fields. For example, if TEMPLATE_NAME is set to 1BPJ_A, other template-specific fields are given as 1BPJ_A_STRUCT_FILE, 1BPJ_A_ALIGN_FILE, and so on.The chain to be used is specified by an underscore and letter suffix: for example, for 1BPJ_A, chain A will be used. The chain suffix must be included in the label.
*template*_NUMBER *n*	Required. Used to identify a given template in the COMPOSITE_ARRAY (see below). Numbering goes from 1 to 9.

Table 11.3. Keywords for the bldstruct script (Continued)

Keyword syntax	Description
*template*_STRUCT_FILE *filename*	Required. The PDB file to be used for this particular template.
USE_SYMMETRY {true|false}	Turn on or off use of the crystallographic symmetry of the template file when constructing the homology model. Default: false.

Files

In addition to the input files (structure and alignment) and the job files (command input file, *jobname*.inp, and log file, *jobname*.log), bldstruct produces the following output files:

jobname-out.mae Built model in Maestro format.

jobname-out.pdb Built model in PDB format.

Return Value and Errors

bldstruct returns a value of 0 on successful completion, 1 otherwise. All error messages are printed to the file *jobname*.log. There is no comprehensive listing of possible errors as these can arise in a variety of programs and scripts.

Examples

The following is a sample input file for building on two templates:

```
QUERY_OFFSET       0
1TSV_NUMBER        1
1TSV_STRUCT_FILE   pdb1TSV.ent
1TSV_ALIGN_FILE    1TSV.align
TEMPLATE_NAME      1BPJ_A
1BPJ_A_NUMBER      2
1BPJ_A_STRUCT_FILE pdb1BPJ_A.ent
1BPJ_A_ALIGN_FILE  1BPJ_A.align
1BPJ_A_HETERO_0    UMP :317
1BPJ_A_HETERO_1    ZN :35
TAILS              0
SIDE_OPT           true
MINIMIZE           true
KEEP_ROTAMERS      true
MAX_INSERTION_SIZE 1000
BUILD_DELETIONS    true
BUILD_TRANSITIONS  true
USE_SYMMETRY       false
COMPOSITE_ARRAY 11111111111111111111112222222222222222222222222222222222221111111
```

The alignment file format is as follows. The first line corresponds to the query sequence (indicated by `ProbeAA:`) and the second line corresponds to the template sequence (indicated by `Fold AA:`). A period is used as a gap character, and an `X` is used for non-standard residues. Below is a sample alignment file.

```
ProbeAA: .AAEEKTEFDVILKAAGANKVAVIKAVRGATGLGLKEAKDLVESA...PAALKEGVSKDDAEALKKALEEAG
AEVEVK
Fold AA: AAQEEKTEFDVVLKSFGQNKIQVIKVVREITGLGLKEAKDLVEKAGSPDAVIKSGVSKEEAEEIKKKLEEAG
AEVELK
```

The sequences are split for formatting reasons, and should be on a single line in the actual file.

11.3 fr—Fold Recognition

`fr` finds template proteins that are similar to the query sequence at both sequence and structure levels.

The fold recognition program used in Prime differs from standard sequence search program such as BLAST by taking into account secondary structure matching and profile-sequence matching. As a result, `fr` can find structural homologs with low sequence identity to the query (<15-20%), which makes it a desirable tool when a given query does not have any high sequence identity homologs in PDB.

Templates used by `fr` are stored in Prime's internal fold library, which is a non-redundant subset of the PDB database and is updated periodically. Individual domains of each multi-domain template are also included in the fold library. When selecting homologs for a query sequence, a scoring function takes into account profile-sequence matching, secondary structure matching and length compatibility between the query sequence and a template, for either a full chain entry or a domain entry. Templates are ranked based on this score.

In this release, we have improved the accuracy of fold recognition with a new set of PSI-BLAST parameters for more accurate profile-sequence matching and with increasing accuracy in secondary structure predictions. As in the previous releases, the top 25 templates are automatically selected in Maestro after Fold Recognition for the next step in the Threading path, Build Backbone. However, because of the enhancements, we now recommend selecting only the top 20 templates if the query is mainly beta according to the secondary structure prediction. For all other secondary structure types, select the top 10 if the query sequence is short (<=120 residues) or the top 5 if the query sequence is longer.

Please note that at least one secondary structure prediction of the query sequence has to be provided in order to run `fr`. `fr` does not predict secondary structure itself.

Syntax

```
$SCHRODINGER/fr jobname [options]
```

Options

The `-DEBUG` and `-HOST` options are recognized by `fr`. See Section 2.3 of the *Job Control Guide* for more information.

Command Input File

The command input file is named *jobname*`.inp`. This input file contains keyword-value pairs, one pair per line separated by one or more spaces. The keywords are given in Table 11.4.

Table 11.4. Keywords for the fr script.

Keyword syntax	Description
`QUERY_NAME` *name*	Name of the query sequence
`QUERY_FILE` *filename*	Query sequence file, in FASTA format
`ALIFORMAT` *value*	Select format of alignments file. Default is Maestro format. For "normal" format set *value* to `dev`.
`BGNRES` *n*	The first residue of the query that is used in search for homologs. Default: 1 (the first residue).
`ENDRES` *n*	The last residue of the query that is used. Default: the last residue of the complete query
`PREDSS`[*_i*] *filename*	Secondary structure prediction file of the query sequence, in CASP format. One occurrence of this keyword is required for each file, and there must be at least one SSP. If only one prediction is used, the keyword is PREDSS; if more predictions are used, the keyword is suffixed with an index, starting from zero: `PREDSS_0, PREDSS_1`, and so on.

Files

- Input file—named *jobname*`.inp`. Contains keywords for running the program.
- Query sequence file—File containing the complete sequence of the query, in FASTA format. Specified in the input file.
- Secondary structure prediction files in CASP format for the query sequence. See below for an example of CASP format.
- Output Ranking File—File named *jobname*`.out` containing a list of templates ranked according to the FR scoring function as described above.

- Output alignment file—File named *jobname*`.ali` containing pairwise alignment between the query and a template. By default, the alignment is in Maestro format (m2io). In order to view alignments in normal format (i.e. aligned residues from query and template are placed on top of each other), add the keyword `ALIFORMAT` to the input file with value set to `dev`.
- Log file—File named *jobname*`.log` containing progress of the program, including warnings and error messages.

Return Value and Errors

The script returns 0 for success and 1 for failure.

In case of failure, check the log file for details on what might be have caused the program to fail. To turn on debugging output use the `-DEBUG` command line option as follows:

```
$SCHRODINGER/fr -DEBUG jobname
```

Examples

The following file demonstrates the use of three SSPs.

```
QUERY_NAME      HypProtein_furiosus_truncated
QUERY_FILE      HP.seq
PREDSS_0        prime_foldrecog_run1_HP-ssp1.casp
PREDSS_1        prime_foldrecog_run1_HP-ssp2.casp
PREDSS_2        prime_foldrecog_run1_HP-ssp3.casp
BGNRES          1
ENDRES          197
```

The following is an example of a CASP format file. The CASP format has a minimum of 2 columns per line, with the first being the one-letter code for the amino acid and the second being the secondary structure type. An optional third column gives the confidence level of the prediction.

```
PFRMAT SS
TARGET 1HL6:A
AUTHOR xxxx-xxxx-xxxx
REMARK written by Schrodinger LLC Software
METHOD not applicable
MODEL 1
M C 1.00
A C 1.00
D H 1.00
V H 1.00
...
```

```
E C 1.00
K C 1.00
R C 1.00
R C 1.00
R C 1.00
END
```

11.4 multirefine—Multi-Step Extended Sampling

The multirefine script automates extended sampling protocols, which consist of multiple executions of `refinestruct` on a set of structures. Multirefine handles the submission of the subjobs and the compilation of the results. Currently available protocols include extended loop sampling, ultra-extended loop sampling, and cooperative loop pair sampling.

Syntax

`$SCHRODINGER/multirefine` *input-file* [*options*]

where *input-file* is the name of the command input file, with or without a `.inp` extension, and serves also as the default job name. The input file must be either the first or last command-line argument, and everything else on the command line is passed as is to Job Control.

Options

The standard Job Control command line options listed in Table 11.1 are accepted. There are no options specific to this program.

Command Input File

Most of the keywords are the same as the general keywords and the keywords for protocol 1 (loop and helix refinement) for `refinestruct`: see Section 11.5 on page 98 for details. The differences and new keywords are described below.

Table 11.5. Keywords for the multirefine input file

Keyword syntax	Description
`HOST` *hostname*	Job host to which the subjobs will be submitted. The top-level driver script will run on the host specified by the `-HOST` command-line argument (or on the launch host if none is specified.) If *hostname* is the same as that specified by `-HOST` or is `localhost`, the subjobs will run on the same host as the top-level driver (or be submitted to the same queuing system if the top-level host is a queue). Otherwise, the driver will submit subjobs using *hostname* as their command-line argument. Not available with `refinestruct`.

Table 11.5. Keywords for the multirefine input file (Continued)

Keyword syntax	Description
LOOP_*i*_RES_*j*	Specify beginning and end of loops. Same as for refinestruct. Required if SEG_LIST is not given. Depending on the protocol, 1, 2, or an arbitrary number of loops may be specified. For example, to specify the beginning and end of loop 0, from residue 10 to residue 20 in chain A, use the following: LOOP_0_RES_0 A:10 LOOP_0_RES_1 A:20
MAX_JOBS *n*	Maximum number of subjobs that can run at one time. This can be set to a value less than the value of THREADS if sufficient resources are not available. Not available with refinestruct.
PRIME_TYPE *n*	Type of refinement to be carried out. Can be specified as either a number or text string. Different from refinestruct. Available choices are: 0 or default: Equivalent to running a single loop refinement using refinestruct. Not particularly useful except for completeness and testing purposes. Does not recognize THREADS keyword. 1 or extended: Used for Extended Sampling from the GUI. Does not recognize MIN_OVERLAP and MAX_CA_MOVEMENT keywords, as these are used internally by the protocol. Accepts multiple loops as input which will be run sequentially. 2 or long_loop: No longer used. 3 or loop_pair: Used for Cooperative Loop Sampling from the GUI. Exactly two loops must be specified. 2 or long_loop_2: Used for Ultra-extended Sampling from the GUI. Does not recognize MIN_OVERLAP and MAX_CA_MOVEMENT keywords, as these are used internally by the protocol. Also does not recognize THREADS keyword, as the sampling is determined by the loop length. Only one loop can be specified.
STRUCT_FILE	Required. The input structure file in Maestro format. Same as for refinestruct.
SEG_LIST *filename*	Name of a file containing the list of loops to be refined. For the example given for the keyword LOOP_*i*_RES_*j*, the file would consist of the single line: loop A:10 A:20 Not available with refinestruct.
THREADS *n*	Number of simultaneous jobs to be run during selected refinement stages. Determines the sampling level for some protocols. Note: the jobs specified by THREADS do not have to actually run in parallel—see MAX_JOBS. Not available with refinestruct.

Files

In addition to the *jobname*.log and *jobname*.err files, `multirefine` produces the following output files:

jobname`-out.mae`	Final structures, all in a single file.
jobname`.pdb`	PDB format file corresponding to lowest-energy structure. Other structures will have corresponding files *jobname*`-2.pdb`, and so on.
jobname`.restart`	Contains information that allows a job to be restarted if it fails.
jobname`-run-`*N*`.archive.zip`	Archives of subjob output files.

Temporary files, which are normally deleted when no longer needed unless the debugging option is used, are named according to the following patterns:

jobname`-conf-`*N*`.*`	Files associated with temporary structures. All active structures in the ensemble have a unique serial number *N*.
jobname`-run-`*N*`.*`	Files associated with a given subjob. All subjobs have a unique serial number *N*.

Return Value and Errors

`multirefine` returns a value of 0 on successful completion, 1 otherwise. All scripts run by `multirefine` that return a value of 1 also print a message to the file *jobname*`.err`. This file is empty otherwise, as all non-fatal messages are printed to the file *jobname*`.log`. Output from Job Control goes to standard output, and any errors arising before the driver script has started likewise go to standard output or standard error.

Restarting an Incomplete Job

If a job fails, you can restart it by submitting it again from the same directory. The *jobname*`.restart` file and the archives of subjobs must be present in the directory, and the directory must be accessible from the host on which the driver job runs. Also, the job records for the failed jobs must not be purged from the job database, so that the restart mechanism has access to information on the failed subjobs. If the job records are missing, the subjobs are repeated.

Failed jobs that were started from Maestro must be restarted from the command line. You can use the `-PROJ` and `-DISP` options to assign the job to the Maestro project and set the incorporation mode.

Once the job finishes successfully, the subjob archives and the restart file are removed.

It is not necessary to use -LOCAL or -SAVE to ensure that a job can be restarted, but there must be sufficient space in the working directory for the subjob archives.

Examples

The following is an example of an input file produced by Maestro for a Serial loop sampling (Extended Medium) job:

```
STRUCT_FILE          prime_refine.mae
PRIME_TYPE          1
LOOP_0_RES_0        _:35
LOOP_0_RES_1        _:40
THREADS             4
MAX_JOBS            2
NUM_OUTPUT_STRUCT   4
RES_SPHERE          7.50
USE_CRYSTAL_SYMMETRY        false
USE_RANDOM_SEED     true
SEED                0
HOST                localhost
```

Setting PRIME_TYPE to 1 selects the extended sampling protocol; setting THREADS to 4 corresponds to the Medium sampling level. Setting MAX_JOBS to 2 requires that no more than 2 subjobs can run at once (e.g. if using a 2-processor machine.)

Environment Variables

In addition to the standard Schrödinger environment variables, multirefine recognizes the following environment variables:

PSP_RB_DEBUG PSP_DEBUG	If this environment variable is set, debugging will be turned on and the subdirectory containing intermediate files will be kept. Equivalent to using the command-line option -DEBUG.
PSP_RB_LOCAL PSP_LOCAL	If this is set, all calculations will run in the local launch directory. Equivalent to using the command-line option -LOCAL.
TFM_QUIET TFM_VERBOSE	These environment variables can be set to decrease or increase the job-related output in the log file.

11.5 refinestruct—Refine a Protein Structure

The `refinestruct` script runs protein structure refinement jobs (side chain prediction and energy minimization as well as loop and helix prediction) using the Protein Local Optimization Program (PLOP).

`refinestruct` handles all the tasks associated with the refinement of protein structures via PLOP: parsing and preparation of input files, interaction with PLOP, and handling of output structures, etc. It also fixes the structural issues that are fixed by `primefix.py` (see Section 12.3 on page 118). All jobs handled by `refinestruct` are run under Schrödinger Job Control.

Multithreaded execution (with Open MP) can be enabled for `bldstruct` jobs by setting the environment variable `OMP_NUM_THREADS`. It is recommended to set this environment variable in the `schrodinger.hosts` file for hosts on which multithreading is supported—see Section 6.1 of the *Installation Guide*.

Syntax

```
$SCHRODINGER/refinestruct [options] jobname
```

The input is read from the file *jobname*`.inp` and the structural output is written to the file *jobname*`-out.mae`. If the input structure file has alternate coordinates, the refinement is performed on the first set of alternate coordinates.

Options

The `refinestruct` script accepts the options listed in Table 11.1.

Command Input File

The command input file contains a list of keywords that control the operation of the `refinestruct` script, one per line. The keywords are listed in the tables below. Generally valid keywords are listed in Table 11.6 and protocol-specific keywords are listed in Table 11.7, under the protocol to which they apply.

Table 11.6. General keywords for the refinestruct script

Keyword syntax	Description
`ECUTOFF` *value*	Energy cutoff for return of conformations. Returns all conformations within *value* kcal/mol of the lowest-energy structure. It can be used in conjunction with `NUM_OUTPUT_STRUCT`, to impose a maximum on the number of such conformations to return. As with `NUM_OUTPUT_STRUCT`, it only applies to single loop refinements.

Table 11.6. General keywords for the refinestruct script (Continued)

Keyword syntax	Description
EXT_DIEL *value*	Dielectric constant of the continuum solvation model. Default: 80.0. Used with SGB_MOD sgbnp.
INT_DIEL *value*	Dielectric constant used within the radius of an atom. Default: 1.0.
NUM_OUTPUT_STRUCT *n*	Number of conformations to return for single loop refinements. This has no effect on helix refinements or jobs composed of more than one refinement (e.g. 2 loops, 1 helix + 1 loop, etc.). All structures are returned in a single structure file in Maestro format. Default: 1.
PAIR_CONSTRAINT_*i* *atom1* , *atom2* , *value*	Specify a constraint on a pair of atoms. The format of the atom specifications is given at the beginning of this chapter. The value is the distance between the atoms in angstroms.
PRIME_TYPE *type*	Type of refinement to perform. The available options are: SIDE_PRED Predict Side Chains (prefix SC) LOOP_BLD Predict Loops and Helices REAL_MIN Minimize Energy (prefix MINI) SITE_OPT Active Site Minimization (including ligand) ENERGY Energy Calculation
RESIDUE_*i* *spec*	Specify a residue to refine. The numbering *i* begins at 0, and increases incrementally. The format of the residue specification *spec* is given at the beginning of this chapter.
RETAIN_CORRECTIONS {yes | no}	Retain corrections made when duplicate residue numbers are found. The duplications detected are between non-protein residues or between protein and non-protein residues. The non-protein residue is assigned a new, unique chain name temporarily. If you set this keyword to yes, the new chain name is written to the output file. Default: no.
SEED *n*	The integer to use as the seed for the random number generator if USE_RANDOM_SEED is false. If it is true, SEED is ignored. Default: -1, meaning that the program will generate a random seed rather than use the supplied seed.
SELECT *string*	Specify the manner for specifying which residues to refine. Available options are: all Refine all residues pick Refine the specific residues, indicated by included RESIDUE_*i* parameters (see above). file Read list of residues to refine from a file. The file consists of a series of lines with a residue specification on each line.

Table 11.6. General keywords for the refinestruct script (Continued)

Keyword syntax	Description
SGB_MOD {sgbnp| vdgbnp}	Specify the implicit solvation model to use. The choices are: sgbnp—use the standard generalized Born model. vdgbnp—use the variable-dielectric generalized Born model, which uses different dielectric constants based on the polar or charged nature of the protein residue: 4 for Lys, 3 for Glu/Arg, 2 for Asp/His, 1 for all others. This method improves loop predictions. Default: vdgbnp.
STRUCT_FILE *filename*	Input structure file in Maestro format.
USE_CRYSTAL_SYMMETRY {yes|no}	Set to true if the input structure contains unit cell information and crystal symmetric atoms are to be included in calculation. Default: no.
USE_RANDOM_SEED {yes|no}	Indicates whether to use a random seed for the random number generator. This keyword has no effect on Minimization tasks. Default: no.
USE_MAE_CHARGES {yes|no}	Indicates whether to use atomic partial charges from the Maestro input file for untemplated residues (ligands, cofactors). Default: no.
USE_MEMBRANE {yes|no}	Indicates whether to use the implicit membrane model. The model must be set up from the Setup Membrane panel in Maestro. Default: no.
USE_SGB {yes|no}	Use the SGB implicit solvation model. If the model is not used, calculations are run in vacuum with a dielectric of 6.0. Default: no.

Table 11.7. Protocol-specific keywords for the refinestruct script

Keyword syntax	Description
LOOP_BLD	
CA_CONSTRAINT_*i* *spec*,[*x*,*y*,*z*]*value*	Specify a constraint on the position of a specific alpha carbon atom. The format of the residue specification *spec* is given at the beginning of this chapter. The optional coordinates specify the target position; if omitted, the target position is the initial position. The value is the maximum distance that the C-alpha atom can move from the target position, in angstroms. Specifying a target position allows you to move a loop to a desired location.
LOOP_*i*_RES_*j* *spec*	Specify residue *j* for defining loop *i*. Loops are numbered sequentially, beginning at 0. Two occurrences of this keyword are required, with *j*=0 (the beginning of the loop) and *j*=1 (the end of the loop). *spec* is a residue specification as defined above.

Table 11.7. Protocol-specific keywords for the refinestruct script (Continued)

Keyword syntax	Description
HELIX_BUILD *spec1* / *spec2*	Build the specified sequence as a helix. The format of the residue specifications *spec1* and *spec2* is given at the beginning of this chapter. You should include enough residues on either side of the helical region in the loop prediction to ensure that sufficient flexibility is available to build the loop.
MAX_CA_MOVEMENT *value*	Specify the maximum distance that any alpha carbon atom in the loop should be allowed to move during refinement. Omitting this parameter indicates that no restriction should be applied.
RES_SPHERE *value*	All side chains that lie within this distance of the loop will also be optimized during refinement. This allows these nearby side chains to "react" to the loop being predicted. Default 0.0.
MIN_OVERLAP *value*	Fraction of van der Waals distance used to define a clash. Smaller values allow structures with worse potential atomic overlaps to be generated. Default: 0.70
SITE_OPT	
LIGAND *string*	Residue specification for the ligand.
NPASSES *n*	Number of passes through side chain optimization of the protein residues followed by minimization of the protein residues (including the backbone). These two steps are repeated the specified number of times before proceeding to optimization of the protein and the ligand. Default: 2. Files written from Maestro have NPASSES set to 1.

Files

In addition to the input structure file, defined by the value of STRUCT_FILE, and the job files (the command input file *jobname*.inp, and the log file *jobname*.log), bldstruct produces the following output file:

jobname-out.mae Final structures, all in a single file.

Return Value

refinestruct returns a value of 0 on successful completion, 17 for license errors, and 1 otherwise. All error messages are printed to the file *jobname*.log. There is no comprehensive listing of possible errors as these can arise in a variety of programs and scripts.

Environment Variables

In addition to those used for all Schrödinger software, `refinestruct` uses the environment variable `PSP_OPLS_VERSION` to define the OPLS force field version. The default value is `2005`. It can be set to `2001` to use the earlier set of force field parameters for ligands and non-standard residues. The `OMP_NUM_THREADS` environment variable is also supported for OpenMP multithreaded execution (see above).

Examples

The following is a sample minimization input file:

```
STRUCT_FILE       input-structure.mae
PRIME_TYPE        REAL_MIN
SELECT            pick
RESIDUE_0         A:1
RESIDUE_1         A:20
RESIDUE_2         A:23
RESIDUE_3         A:26
RESIDUE_4         A:27
RESIDUE_5         A:30
RESIDUE_6         A:31
RESIDUE_7         A:36
RESIDUE_8         A:40
RESIDUE_9         A:42
RESIDUE_10        A:53
USE_RANDOM_SEED   false
SEED              0
```

The arrangement in columns is for ease of reading only—the spacing is not significant.

11.6 ska—Protein Structure Alignment

The SKA program carries out structural alignment of proteins by aligning protein backbones at different levels of detail.

The `ska` script provides an interface to the SKA structural alignment program. As with other computational programs in Prime its operation is controlled through simple input files. The allowed keywords and their values are discussed in more detail below. The `ska` script allows you to run the SKA program in three different modes under Job Control:

- **align mode**—SKA aligns two or more protein structures to one another.
- **dbcreate mode**—SKA creates databases for use with scan mode. You should split PDB files with multiple chains into single-chain files before adding them to the database, as

SKA has difficulty aligning to multiple chains.

- **scan mode**—SKA searches a database of protein structures and identifies potential structural analogues. By default a non-redundant database of protein chains from the PDB is searched.

Syntax

```
$SCHRODINGER/ska [options] jobname
```

By default the input is read from the file *jobname*`.inp` and the output is written to *jobname*`.out`.

`ska` does not generate output other than the explicitly defined database and a log file in `dbcreate` mode—see below for details.

Options

The only option accepted by `ska` is `-DEBUG`, which turns on debugging mode for both Job Control and the SKA program.

Command Input File

The input file consists of lines containing keyword-value pairs, one per line. The keywords are described in Table 11.8. The set of allowed and required keywords depends on the mode, which is determined by the `MODE` keyword.

Return Value and Errors

The script returns 0 for success and 1 for failure.

In case of failure, check the log file for details on what might be have caused the program to fail. To turn on debugging output use the `-DEBUG` command line option as follows:

```
$SCHRODINGER/ska -DEBUG jobname
```

Environment Variables

The `ska` program handles all required environment variables internally and requires only that the environment variable `SCHRODINGER` be set. Should the environment variable `TROLLTOP` exist in the current environment it will be overwritten.

Table 11.8. Keywords syntax for ska input file

Keyword syntax	Mode	Description
`MODE` *mode*	all	The SKA operating mode; allowed values are: `*align*`: structurally align two or more protein structures to on another. This is the default mode `*scan*`: given a query structure scan a database of protein structures and identify potential structural homologues `*dbcreate*`: create a database that can be search in scan mode from a list of PDB files
`QUERY_FILE` *filename* [*name*]	`*align*` `*scan*`	Required. The input query structure. In `*align*` mode, the structure against which other proteins are aligned. In `*scan*` mode, the structure used for scanning the database. The optional *name* (`*align*` mode only) sets the name of this structure to a value other than the filename (the default). The name is displayed in the output file (see Examples, below). *name* cannot contain the file separator character ('/' on UNIX).
`TEMPLATE` *filename* [*name*]	`*align*`	Required. PDB structures that are to be aligned to the query structure, set by `QUERY_FILE`. The `TEMPLATE` keyword can appear multiple times in order to be able to carry out multiple structure alignments (aligning more than one template structure to the query structure). The optional *name* sets the name of this entry to a value other than the filename (the default). The name is displayed in the output file (see Examples, below). *name* cannot contain the file separator character ('/' on UNIX).
`DATABASE` *filename*	`*scan*`	The absolute path to the database to scan with the query structure. The database can be generated in dbcreate mode. Default: database supplied with Prime.
`DBNAME` *filename*	`*dbcreate*`	The name of the file in which to store the database create by `ska`. *filename* must be the file name without any path specification.
`FILELIST` *filename*	`*dbcreate*`	A file containing a list of absolute filenames for structures (in PDB format) that are to be included in a database of templates that can be search with `ska` in scan mode. Absolute filenames are required to avoid having to copy large numbers of PDB files to the temporary directory in which the job is run.
`GAP_OPEN` *value*	`*align*` `*scan*`	Gap initiation penalty.
`GAP_DEL` *value*	`*align*`	Deletion penalty.

Table 11.8. Keywords syntax for ska input file (Continued)

Keyword syntax	Mode	Description
OUTPUT_FILE *filename*	*align* *scan*	File in which to store the structural alignment. Default is *jobname*.out.
ALISTRUCS_OUTFILE *filename*	*align*	File in which to store the 3D coordinates of the aligned structures.
PSD_THRESHOLD *value*	*align* *scan*	PSD threshold for inclusion in the MSA. Overridden by RECKLESS.
SSE_MINSIM *value*	*align* *scan*	Minimum Secondary Structural Element (SSE) similarity for the alignment.
SSE_MINLEN *value*	*align* *scan*	Minimum SSE length for the alignment.
SSE_WINLEN *value*	*align* *scan*	Align only SSE windows of length *value*.
SWITCH [true|false]	*align* *scan*	Carry out global alignment first and then align subwindows if no similarity was found.
ORDER [seed|psd]	*align*	psd: order the alignment by PSD to the seed structure.
CLEAN	*align*	Exclude structures with a PSD < PSD_THRESHOLD from the alignment.
RECKLESS [true|false]	*align* *scan*	Force the alignment of structures even if no similarity is found.

Examples

The following minimal input file (assumed to be called align.inp) structurally aligns two protein structures to one another:

```
QUERY_FILE query.pdb
TEMPLATE template.pdb
```

where query.pdb is the name of the PDB file containing the query structure (the structure to which we want to align) and template.pdb contains the template structure (the structure we want to align to the query).

When run using $SCHRODINGER/ska align it should produce the following output:

- align.out—a file containing the structural alignment
- align.log—a log file that contains log information and diagnostics on the run

Note that the actual 3D coordinates of the superposed structures are not written out by default You must use ALISTRUCS_OUTFILE to write out the structures.

The following input file (align.inp):

```
QUERY_FILE query.pdb
TEMPLATE template.pdb
ALISTRUCS_OUTFILE superposition.pdb
```

saves the 3D coordinates of the aligned structures in the file superposition.pdb.

The following is a sample input file for database creation in dbcreate mode:

```
MODE dbcreate
FILELIST filelist.dat
DBNAME mydatabase.dat
```

The file filelist.dat contains the file names for the templates, including the path, as follows:

```
/scr/templates/template-1.pdb
/scr/templates/template-2.pdb
/scr/templates/template-3.pdb
/scr/templates/template-4.pdb
```

A minimal input file for a scan is as follows:

```
MODE scan
QUERY_FILE query.pdb
```

Running ska with this input file would scan the database supplied with Prime for structural analogs. The output from the structural search contains pairwise alignments between the query structure and every structure in the database (assuming the two structures are sufficiently similar). The results can be analyzed in a more convenient way with the graphical SKA Results Viewer utility, $SCHRODINGER/utilities/SkaResultsViewer.

The use of the optional *name* argument is illustrated next. For the following input file:

```
QUERY_FILE query.pdb 1ctf
TEMPLATE 1dd3_A.pdb
```

the output file is as follows:

```
            .........+.........+.........+.........+.........+.........+.....
 1ctf    1                                                              ceeeeeecc
1dd3_A    1  cchhhhhhhhhhhchhhhhhhhhhhhhhhhcccchhhhhhhhhhhhhhhhhhhhhhhhcccceeeeeecc
 1ctf    1  ------------------------------------------------------------EFDVILKAA
1dd3_A    1  MTIDEIIEAIEKLTVSELAELVKKLEDKFGVTAAAPVAVAAAPVAGAAAGAAQEEKTEFDVVLKSF
```

```
            ..........+.........+.........+.........+.........+.........+.
 1ctf    62  ccchhhhhhhhhhhhccchhhhhhhhhhc c  eeeeeccchhhhhhhhhhhhhhcceeeee
1dd3_A   67  ccchhhhhhhhhhhhccchhhhhhhhhhhcccccccceeccchhhhhhhhhhhhhhcceeeee
 1ctf    62  GANKVAVIKAVRGATGLGLKEAKDLVESA-P--AALKEGVSKDDAEALKKALEEAGAEVEVK
1dd3_A   67  GQNKIQVIKVVREITGLGLKEAKDLVEKAGSPDAVIKSGVSKEEAEEIKKKLEEAGAEVELK


RMSD: 0.763733
PSD: 0.024524
```

Without the *name* argument for the `QUERY_FILE` keyword, the output would be

```
.........+.........+.........+.........+.........+.........+.....
query    1                                                          ceeeeeecc
1dd3_A    1  cchhhhhhhhhhhchhhhhhhhhhhhhhhhcccchhhhhhhhhhhhhhhhhhhhhhhcccceeeeeecc
query    1  ------------------------------------------------------------EFDVILKAA
1dd3_A    1  MTIDEIIEAIEKLTVSELAELVKKLEDKFGVTAAAPVAVAAAPVAGAAAGAAQEEKTEFDVVLKSF

            ..........+.........+.........+.........+.........+.........+.
query   62  ccchhhhhhhhhhhhccchhhhhhhhhhc c  eeeeeccchhhhhhhhhhhhhhcceeeee
1dd3_A   67  ccchhhhhhhhhhhhccchhhhhhhhhhhcccccccceeccchhhhhhhhhhhhhhcceeeee
query   62  GANKVAVIKAVRGATGLGLKEAKDLVESA-P--AALKEGVSKDDAEALKKALEEAGAEVEVK
1dd3_A   67  GQNKIQVIKVVREITGLGLKEAKDLVEKAGSPDAVIKSGVSKEEAEEIKKKLEEAGAEVELK


RMSD: 0.763733
PSD: 0.024524
```

11.7 ssp—Secondary Structure Prediction

The `ssp` script provides an interface to the secondary structure prediction programs SSPro, which is bundled with Prime, as well as PSIPRED, which is available from a third party.

Syntax

```
$SCHRODINGER/ssp [options] jobname
```

By default the input is read from the file *jobname*`.inp` and the output is written to *jobname*`.out`.

Options

The standard Job Control command line options listed in Table 11.1 are accepted. There are no options specific to this program.

Command Input File

The command input file consists of lines containing keyword-value pairs, one per line. The keywords are described in Table 11.9.

Table 11.9. Keywords for the ssp input file.

Keyword syntax	Description
`QUERY_FILE` *filename*	Required. Name of the FASTA file containing the query sequence
`METHOD` *program*	Required. Name of the secondary structure prediction that is to be run. Allowed values are `sspro`, `psipred`, `profking`, and `all`. `all` means run all available programs. If the input file contains multiple `METHOD` keywords, only the last one is used.
`OUTPUT_FILE` *filename*	Optional. Name of the output file. Default is *jobname*`.out`.

Return Value and Errors

`ssp` returns 0 if successful and a non-zero value in all other cases.

Environment Variables

The environment variables that can be used to control the secondary structure prediction programs are listed in Table 11.10.

Table 11.10. Environment variables for secondary structure prediction programs.

Environment Variable	Description
`PSP_SSPRO_DB`	BLAST database to be searched when assembling the multiple sequence alignment used in the SSPro prediction
`PSP_PSIPRED_DB`	BLAST database to be searched when assembling the sequence profile used in the PSIPRED prediction

Examples

The following minimal input file, called `sspro.inp`, generates an SSPro secondary structure prediction for the sequence stored in file `query.fasta`:

```
QUERY_FILE query.fasta
METHOD     sspro
```

The prediction produced by the back-end is written to the file `sspro.out`.

The following input file, assumed to be called `all-methods.inp`, instructs the `ssp` script to produce secondary structure prediction for all methods available—SSPro and, depending on the local installation, PSIPRED.

```
QUERY_FILE query.fasta
METHOD all
```

11.8 sta—Single Template Alignment

The Single Template Alignment (STA) program in PRIME is designed for protein sequences with medium to high sequence identity (>25%). STA differs from standard sequence alignment such as BLAST by taking into account secondary structure matching as well as profile-sequence matching. As a result, STA often generates better alignment than BLAST in the regions where sequence conservation is relatively weak. STA can also take in constraints, such as user-selected residue pairs to be aligned together and/or residues in either query or template that are not to be aligned (gaps).

STA can use templates directly from the PDB, as selected by running the Find Homologs step, and can also use templates from the Prime fold library, as selected in the Fold Recognition step. For the latter, you can export the template from the Fold Recognition table, and save it as a PDB file.

Syntax

```
$SCHRODINGER/sta [options] jobname
```

The input is read from the file *jobname*`.inp` and the output is written to *jobname*`.out`.

Options

The `sta` script accepts the `-DEBUG` and `-HOST` options, as described in Table 11.1.

Command Input File

The input file consists of lines containing keyword-value pairs, one per line. The keywords are described in Table 11.11.

Files

- Input file—named *jobname*`.inp`. Contains keywords for running the program.
- Query sequence file—File containing the complete sequence of the query, in FASTA format. Specified in the input file.
- Secondary structure prediction files in CASP format for the query sequence. See page 93 for an example of CASP format.

Table 11.11. Keywords for the sta program.

Keyword syntax	Description
QUERY_NAME *name*	Name of the query sequence.
QUERY_FILE *filename*	File name of the query sequence in FASTA format.
FORMAT *string*	Alignment file format. Default is Maestro. Set to dev for plain text format—see page 111 for an example.
BGNRES *n*	The first residue of the query that is used in alignment. Default: 1.
ENDRES *n*	The last residue of the query sequence that is used. Default: the last residue of the original query sequence.
PREDSS[_*i*] *filename*	Secondary structure prediction file of the query sequence, in CASP format. One occurrence of this keyword is required for each file, and there must be at least one SSP file specified. If only one prediction is used, the keyword is PREDSS; if more predictions are used, the keyword is suffixed with an index *i*, starting from zero: PREDSS_0, PREDSS_1, and so on. For an example of CASP format, see page 93.
TEMPLATE_PDB *filename*	Filename of the template structure, in PDB format.
MODE *string*	Optional. Needed in the input file only when there are user-specified constraints. In that case, the value is set to user.
PAIR *pair-list*	Optional. User-specified residue pairing, used only when MODE is set to user. All the residue pairs can be added after the keyword in one or multiple lines. For example, the following lines pair residue 12, 20, 60 from the query sequence with residue 15, 24, 65 from the template. MODE user PAIR 12 15 20 24 PAIR 60 65
GAP *gap-list*	Optional. User-specified gaps, used only when MODE is set to user. Can be used with or without PAIR. Gaps are specified by the residues in either query (P) or template (T) that are aligned with them. For example, the following opens gaps against query residue 10, 11, template residue 34 and 35. MODE user GAP P10 P11 T34 T35

- Template PDB file—Template structure file, in PDB format. The sequence used for the template is taken from SEQRES lines in the PDB file. If SEQRES is not found in the PDB file, the template sequence is then taken from ATOM records.
- Output alignment file—File named *jobname*.out containing pairwise alignment between the query and the template. By default, the alignment is in Maestro format. To generate plain text format, include FORMAT dev in the command input file. In plain text format,

aligned residues from the query and the template are placed on top of each other—see below for an example.

- Log file—File named *jobname*`.log`containing progress of the `sta` program, including warnings and error messages.

Return Value and Errors

`sta` returns 0 for success and non-zero for failure. In case of failure, check the log file for details, and search for `ERROR`. To further analyze the failure, use the `-DEBUG` option when you run `sta`.

Examples

Below is an example input file, where `query_seq.fasta` stores the query sequence in FASTA format.

```
QUERY_NAME      query
QUERY_FILE      query_seq.fasta
PREDSS_0        query-ssp1.casp
PREDSS_1        query-ssp2.casp
PREDSS_3        query-ssp3.casp
BGNRES          1
ENDRES          149
TEMPLATE_PDB    pdb1vpe.ent
```

Below is an example of the plain text format for output alignment files. This format is generated when you include `ALIFORMAT dev` in the command input file.

```
probe length=70======tmplt_T0281_d1dq3a2 length=79 TotalScore= 42 SEQ= -4 SSTR= 102
 SEQID= 0.061538 nali= 65
probe bgnRes=1 endRes=70    template bgnRes=3 endRes=75

ProbeAA: ..MWMPPRPEEVARKLRRLGFVERMAKGGHRLYTHPDGRIVVVPFH...SGELP....KG
ProbeSS:   LLLLLLHHHHHHHHHHLLLEEEEELLLEEEEELLLLLEEEeLLL   LLLLL    HH
Fold AA: GNFGLPLNFNAFKEWASEYGVEFKTNGSQTIAIIND...ERISLGQWHTRNRVSKAVLVK
Fold SS: LLLEELLLHHHHHHHHHLLLLEEEEELLEEEEEELL   EEEELLLHHHHLLEEHHHHHH

ProbeAA: TFKRILRDAGLTEEEFHNL....
ProbeSS: HHHHHHHHhLLLHHHHHLL
Fold AA: MLRKLYEATK.DEEVKRMLHLIE
Fold SS: HHHHHHHHHL LHHHHHHHHHHL
```

The file contains header information at the top, then in the alignment section, the query sequence (`ProbeAA`), query composite secondary structure (`ProbeSS`), template sequence (`Fold  AA`) and template secondary structure (`Fold  SS`) are given. Gaps are denoted by periods in the sequences and by spaces in the secondary structures.

11.9 tfm—Tertiary Folding Module

The tfm program generates one or more complete backbone conformations starting from a primary sequence and secondary structure, along with optional domain boundaries and sheet topologies. In addition, `tfm` can accept several different types of constraint including multiple coordinate templates, multiple types of distance constraints, and dihedral angle constraints. A large number of run-time parameters allow the code to be used in a variety of ways, from minimizing a homology model to full ab initio folding.

Syntax

The `tfm` program has hundreds of user-specified options contained in over a dozen input files. A detailed description of all the options is beyond the scope of this document. The `refine` command can be used as a wrapper to prepare default input files. If suitable input has been prepared, there is also a top-level wrapper for `tfm` that can be used to run the program directly under Schrodinger Job Control. It can be invoked from the command line via:

`$SCHRODINGER/tfm` *input-file* [*options*]

where *input-file* is typically the job name, with or without a `.inp` extension. The input file must be either the first or last command-line argument, and everything else on the command line is passed as is to Job Control.

Options

All options are specified in the input file, and there are no command-line options, other than those recognized by Job Control.

Command Input File

There are only three keywords recognized in the command input file, listed in Table 11.12.

Table 11.12. Keywords for tfm program.

Keyword	Description
`INPUT_FILE` *filename*	The input file passed to the executable.
`OUTPUT_FILE` *filename*	The output produced by the executable.
`STRUCT_NAME` *name*	Optional. Prefix for files in the local directory that will be automatically treated as input, such as template or constraint files.

Files

In addition to the descriptive output in the file specified by OUTPUT_FILE and the error messages in *name*.err, tfm produces the following files, where the first part of the file name is tfm by default but can be changed by other input options, and *N* is a serial number.

tfm.*N*.pdb	Output structures, if applicable.
tfm.*N*.spair	Possible strand pair assignments for unpaired strands, if automatic strand-pairing is being used.

Return Value and Errors

tfm returns a value of 0 on successful completion, 1 otherwise. All error messages are printed to the file *name*.err, where *name* is either the STRUCT_NAME value or the SEQNAME parameter in the file specified by INPUT_FILE. This file is empty otherwise, and all output from the program itself is written to the file specified by OUTPUT_FILE. Output from Job Control goes to stdout, and any errors arising before the driver script has started will likewise go to stdout or stderr.

Examples

Sample input for tfm can be generated by running refine with the -LOCAL and -DEBUG options. The command

```
refine sample
```

creates a directory run_*sample* with the input file *sample*.1.in (among other files determined by the refinement type). An input file *sample*_tfm.inp containing the following lines can be used to run the job from the command line

```
INPUT_FILE sample.1.in
OUTPUT_FILE sample.1.out
```

with the command

```
tfm sample_tfm
```

Environment Variables

In addition to the standard environment variables tfm recognizes the following environment variables:

TFM_QUIET TFM_VERBOSE	These environment variables can be set to decrease or increase the job-related output in the log file. The value to which they are set is irrelevant.

Command-Line Utilities

12.1 Multiple Template Alignment: structalign

If multiple templates are selected in the Find Homologs step, they must be rotated into the same coordinate reference frame before they can be used. The Align Structures button is the usual means of performing structural alignment of templates, but the `structalign` utility can be run from the command line.

The input files must be in PDB format and the templates in them should consist of single chains. To use the output files, each must be imported into Find Homologs and selected as one of the templates, replacing the original, unaligned version.

Syntax:

`$SCHRODINGER/utilities/structalign [`*options*`]` *input-files*

The input files must be in PDB format. The output file name is generated by adding the prefix `rot-` to the input file name.

Options:

`-h`	Print usage message.
`-force`	Force alignment even if structures are not sufficiently similar.

To use `structalign` from the command line:

1. Obtain the PDB structure file of each template. This can be done easily by clicking on the sequence name of each template of interest to display it in the Workspace.
2. For each structure, choose Export from the Project menu of the Maestro main window to save the coordinates of the structure to a file. Select PDB format.

   You should now have the following files:

   template1`.pdb` *template2*`.pdb` *template3*`.pdb`
3. At the command line, enter:

   `$SCHRODINGER/utilities/structalign` *template1*`.pdb` *template2*`.pdb` *template3*`.pdb`

This command runs the `structalign` utility and returns one aligned file for each template:

`rot-`*template1*`.pdb`
`rot-`*template2*`.pdb`
`rot-`*template3*`.pdb`

Note: If you encounter a problem involving templates with multiple chains, use the `getpdb` utility (see Section D.2.3 of the *Maestro User Manual*) to extract each chain into a separate `.pdb` file, then run `structalign` again.

4. Import each `rot-`*template*`.pdb` file using the Import button.

Once the files are imported and the aligned templates appear in the Homologs table, select them all using shift-click or control-click. (Do not select the original structures.) Click Next to continue to the Edit Alignment step.

12.2 Format Conversion: seqconvert

This utility converts sequences between any permitted input file format and output file format. This may be useful if autoconversion does not proceed as expected.

Syntax:

`$SCHRODINGER/utilities/seqconvert [`*options*`]`
input-format input_filename output-format output_filename

One use of the `seqconvert` utility is to convert SSPs produced in Prime to FASTA format with the following command:

`$SCHRODINGER/utilities/seqconvert -inative` *infile*`.seq`
`-ofasta` *outfile*`.fasta`

Options are described in Table 12.1.

Table 12.1. Keywords for the `seqconvert` *utility*

Option	Description
`-start`	Read from this sequence in the input file
`-end`	Read to (and including) this sequence in the input file
`-h`	Display usage message
`-total`	Read at most this number of sequences

Table 12.1. Keywords for the `seqconvert` *utility (Continued)*

Option	Description
`-ali`	Read an alignment in the input file
`-pdb_use_atoms`	Use ATOM (not SEQRES) records with PDB files
Input File Formats:	
`-iany`	Autodetect input file format
`-inative`	Input file in native (Schrödinger) format
`-ifasta`	Input file in FASTA format
`-ipdb`	Input file in PDB format
`-inbrf_pir`	Input file in NBRF-PIR format
`-itext_plain`	Input file in TEXT-PLAIN format
`-igcg`	Input file in GCG format
`-iswissprot`	Input file in SWISS-PROT format
`-igenbank`	Input file in GENBANK format
`-incbi`	Input file in NCBI format
`-iphylip`	Input file in PHYLIP format
`-inexus_paup`	Input file in NEXUS-PAUP format
`-iselex`	Input file in SELEX format
`-istaden`	Input file in STADEN format
`-ipfam_stockholm`	Input file in PFAM-STOCKHOLM format
`-icasp`	Input file in CASP format
Output File Formats:	
`-onative`	Output file in native (Schrödinger) format
`-ofasta`	Output file in FASTA format
`-onbrf_pir`	Output file in NBRF-PIR format
`-otext_plain`	Output file in TEXT-PLAIN format
`-ogcg`	Output file in GCG format
`-oswissprot`	Output file in SWISS-PROT format
`-ocasp`	Output file in CASP format
`-ogenbank`	Output file in GENBANK format

12.3 Structure Preparation: primefix.py

Python script for preparing structures for use with Prime, which requires certain conventions for residue naming. You should not normally need to use this script, because its functions are performed when you run a Prime refinement job. The script performs the following actions:

- Assigns a unique, single residue name, residue number, and chain name to all ligand residues, and ensures that new name doesn't conflict with amino acid residue names.
- Left-justifies 3-character residue names, and ensures that 1- and 2-character residue names are treated appropriately.
- Converts PDB names to upper case.
- Deletes all bonds to metals and assigns formal charges to the metal and previously attached atoms. FeS clusters are fixed with the appropriate bonds and formal charges.
- Sets water molecule residue names to `"HOH "` and PDB atom names to `" O "`, `"1H "` and `"2H "`. This action is performed for residues named `TIP`, `TIP3`, `TIP4`, `SPC`, `HOH`, and `DOD`.
- Renames terminal caps: `"NME "` to `"NMA "`, pdbname `" C "` of NMA to `" CA "`, and pdbname `" CA "` of ACE to `" CH3"`.
- Atoms from the same molecule are placed together in the structure output.

Prime can fail if a residue is not in order of connectivity. If this is the case in your structure, it should be exported from Maestro as a PDB file, then reimported and exported in Maestro format before running this script.

Syntax:

```
$SCHRODINGER/run primefix.py infile outfile
```

infile	Input file (Maestro format) containing the structure
outfile	Output structure file (Maestro format) containing the prepared structure

12.4 Structure Alignment: align_binding_sites

This script performs a pairwise superposition of multiple structures using the C-alpha atoms of selected residues. The reference structure for the alignment is the first structure in the input file. The C-alpha atoms used for the alignment come from residues within the specified cutoff distance from the ligand in the reference structure. Alternatively, a list of residues may be given for the alignment, which must correspond to residues in the reference structure. C-alpha atoms in the structures that are aligned that are greater than the specified distance from any C-alpha in the reference are not considered in the alignment.

`$SCHRODINGER/utilities/align_binding_sites` [*options*] *input-file*

The options are given in Table 12.2. The Job Control options described in Table 2.1 and Table 2.2 of the *Job Control Guide* are supported, with the exception of `-INTERVAL`.

Table 12.2. Options for the align_binding_sites command.

Option	Description
`-v[ersion]`	Show the program version and exit.
`-h[elp]`	Show usage message and exit.
`-l[igmol]` *molnum*	Molecule number of the ligand. Default is to automatically detect the ligand.
`-c[utoff]` *value*	Binding site cutoff distance from the ligand in angstroms. Residues within this distance of the ligand are used in the alignment. Default: 5.
`-d[ist]` *value*	Atom pairs greater than this distance are not considered in the alignment (default=5)
`-p[realigned]`	Structures are prealigned. Do not run global protein structure alignment (SKA).
`-r[es]` *list*	Residues for the alignment. This option overrides the use of `-cutoff`. Format for a residue is *chain*`:`*resnum*. If there is no chain name, use an underscore, e.g. `_:999` . At least three residues are required. The comma-separated list should not have spaces.
`-j[obname]` *jobname*	Jobname to use. The default is based on the input file name.

12.5 Database Update Utilities

Prime uses two databases, the PDB database and the BLAST database. As these databases are continually being updated, the versions distributed with the software will not have the latest contributions. Utilities have been provided for updating these databases from the web sites.

12.5.1 update_BLASTDB

This utility updates the BLAST database from a copy on the Schrödinger web site that is synchronized nightly with the BLAST web site. The preformatted database files are downloaded.

Syntax:

```
$SCHRODINGER/utilities/update_BLASTDB
```

The BLAST database is installed in the first location found in the following list:

- `$PSP_BLASTDB`
- `$SCHRODINGER_THIRDPARTY/database/blast`
- `$SCHRODINGER/thirdparty/database/blast`

These environment variables are described in Appendix B. You must have permission to write to the destination in order to run this utility.

12.5.2 rsync_pdb

This utility creates or updates a local mirror of the PDB.

Syntax:

```
$SCHRODINGER/utilities/rsync_pdb
```

The PDB is installed in the first location found in the following list:

- `$SCHRODINGER_PDB`
- `$SCHRODINGER_THIRDPARTY/database/pdb`
- `$SCHRODINGER/thirdparty/database/pdb`

These environment variables are described in Appendix B. You must have permission to write to the destination in order to run this utility.

This utility uses an rsync server on the Schrödinger web site that enables the files to be downloaded. The files are written in the new remediated format. The Schrödinger site is updated nightly from the PDB.

Third-Party Programs

A.1 Location of Third-Party Programs

As well as internal and bundled software and data, Prime uses certain external third-party programs and databases for sequence and homolog searches, protein family searches, and secondary structure prediction. These are described under the headings in this topic.

A.1.1 BLAST/PSI-BLAST, Pfam, and the PDB

See the home pages for these programs and servers for more information:

- BLAST/PSI-BLAST

  http://www.ncbi.nlm.nih.gov/blast

  The BLAST version distributed with Prime is version 2.2.16. To download the BLAST database, you should use the `update_BLASTDB` utility. This utility uses rsync to download or update the database, in the correct format for use with Prime.

- HMMER/Pfam

  http://pfam.sanger.ac.uk

  This site has links to mirror sites that may be more convenient for downloads. You can download the Pfam database from ftp.schrodinger.com/support/hidden/prime/Pfam_fs.gz.

- PDB

  http://www.rcsb.org/pdb/

  Note that the PDB format has changed, as of August 1, 2007. You should use the `rsync_pdb` utility to update your local copy of the PDB for use with Prime. Prime uses the new format, and automatically converts files in the old format.

A.1.2 Secondary Structure Prediction

Two distinct third-party secondary structure prediction programs can be used to increase the accuracy of, and get better results from, Prime. One of these programs (SSpro) is already bundled with Prime. The other (PSIPRED) is highly recommended, and may be manually installed by you, at your election, and subject to applicable license terms, if any. At the third-

party web site whose address is given below, you may download the program and then install it. For instructions on how to install third-party programs, see Section 3.7 of the *Installation Guide* (Linux) or Section 4.2 of the *Installation Guide* (Windows), or the Third Party Programs page of our web site.

The PSIPRED home page is located at:

http://bioinf.cs.ucl.ac.uk/psipred/

and contains information about the program, referrals to terms of use and/or license terms (in the README file), and a link to download the program. The link to download PSIPRED is contained in the section of the page labeled "Overview of prediction methods."

Note: PSIPRED is not available on Windows.

A.2 Third-Party Program Use Agreement

By using the links above or the third-party programs, you acknowledge and agree to the following:

PSIPRED is a third party program ("Third Party Program") and may therefore be subject to third party license agreements and fees. We (i.e., Schrödinger, LLC and its affiliates) have no responsibility or liability, directly or indirectly, for Third Party Programs or for any third party Web sites ("Linked Sites") or for any damage or loss alleged to be caused by or in connection with use of or reliance thereon. Any warranties that we make regarding our own products and services do not apply to the Third Party Programs or Linked Sites, or to the interaction between, or interoperability of, our products and services and the Third Party Programs. Referrals and links to Third Party Programs and Linked Sites do not constitute an endorsement of such Third Party Programs or Linked Sites.

Environment Variables

The only environment variable that *must* be set before you can run Prime is `SCHRODINGER`. The following list is presented for your convenience in case you prefer a different location for any Prime-related directories.

General

`SCHRODINGER_THIRDPARTY`: Directory containing third-party software and databases, which can be installed with the installers supplied by Schrödinger. The databases are installed in the `database` subdirectory; the software in the `bin` subdirectory. Each third-party product has its own subdirectory under these subdirectories. The default location if this environment variable is not set is `$SCHRODINGER/thirdparty`.

PDB

`SCHRODINGER_PDB`: Full path to a user-supplied copy of the PDB database. The default for this directory is `$SCHRODINGER/thirdparty/database/pdb` if only `SCHRODINGER` is set; if `SCHRODINGER_THIRDPARTY` is set, the default is `$SCHRODINGER_THIRDPARTY/database/pdb`. The copy of the database must retain the directory hierarchy used by the PDB and must include the following subdirectories to be usable with Prime:

```
data/structures/all/pdb
data/structures/divided/pdb
data/structures/obsolete/pdb
```

On Windows the `data/structures/all` subdirectory is absent (and is not required).

Blast

`PSP_BLAST_DIR`: Location of Blast executables. Default is `$SCHRODINGER_THIRDPARTY/bin/`*platform*`/blast/bin`, with the default for `SCHRODINGER_THIRDPARTY` noted above.

`PSP_BLAST_DATA`: Location of Blast matrices. Default is `$SCHRODINGER_THIRDPARTY/bin/`*platform*`/blast/data`, with the default for `SCHRODINGER_THIRDPARTY` noted above.

`PSP_BLASTDB`: Location of Blast databases. The BLAST databases distributed with Prime are formatted with version 2.2.16. If your BLAST database does not conform to this format, you might experience problems with Prime. Default is `$SCHRODINGER_THIRDPARTY/database/blast`, with the default for `SCHRODINGER_THIRDPARTY` noted above.

HMMER/Pfam

PSP_HMMER_DIR: Location of HMMER distribution (the executables are assumed to be in the binaries subdirectory). Default is $SCHRODINGER_THIRDPARTY/bin/*platform*/hmmer, with the default for SCHRODINGER_THIRDPARTY noted above.

PSP_HMMERDB: Location of Pfam database. Default is $SCHRODINGER_THIRDPARTY/database/pfam, with the default for SCHRODINGER_THIRDPARTY noted above.

PSIPRED

PSP_PSIPRED_DIR: Location of PSIPRED installation (top level directory containing the bin and data directories). Default is $SCHRODINGER_THIRDPARTY/bin/*platform*/psipred, with the default for SCHRODINGER_THIRDPARTY noted above.

PSP_PSIPRED_DB: Sequence database to use for assembling the PSI-BLAST profile. Allowed values are nr and pdb. Default: pdb.

SSPRO

PSP_SSPRO_DB: Sequence database to use for assembling the SSPRO profile. Allowed values are nr and pdb. Default: pdb.

File Formats

C.1 Input File Formats

The input file formats are:

- native (Schrödinger) format
- FASTA format
- PDB format

  Sequences are read from the PDB SEQRES records, if they exist. Otherwise they are read from the ATOM records. If SEQRES records do not exist and residues are missing from the ATOM records, the query sequence will of course contain gaps.

- NBRF-PIR format
- TEXT-PLAIN format
- GCG format
- SWISS-PROT format
- GENBANK format
- NCBI format
- PHYLIP format
- NEXUS-PAUP format
- SELEX format
- STADEN format
- PFAM-STOCKHOLM format
- CASP format

C.2 Output File Formats

The output file formats are:

- native (Schrödinger) format
- FASTA format
- NBRF-PIR format
- TEXT-PLAIN format
- GCG format
- SWISS-PROT format
- CASP format
- GENBANK format

Error Messages and Warnings

D.1 Input Sequence

Invalid File

This message appears if the system fails to extract a valid sequence from the file.

- Check that the sequence is in one of the supported sequence file formats.
- If FASTA format is used, make sure there are no extra spaces between the ">" and the sequence name, and that there are no extra spaces at the end of the line.

No Sequences Found

This message appears if the system fails to extract any sequences from the workspace.

- Check the Project Table to ensure the correct entries are included in the Workspace.
- Use Open Project in the Maestro Window to open a project.

Invalid Sequence Characters

This message appears if the sequence contains non-standard one-letter code for amino acids.

- Check that sequence uses the one-letter code and is all uppercase.
- Change/edit any non-standard characters from your sequence using an editor, and then restart job.

D.2 Find Homologs

D.2.1 Error Messages

Multiple Templates Not Aligned

- If you select multiple templates and proceed to Edit Alignment, you are reminded that the templates must be structurally aligned to one another.

If one of the following messages appears when you click Next to go to the Edit Alignment step, you will not be able to proceed along the Comparative Modeling path until the problem is resolved:

No Homologs Found

- You may change your search options and continue searching, or proceed to Fold Recognition.

Invalid File

- Check to ensure the homolog file exists in either of the following locations:
  `$SCHRODINGER/thirdparty/database/pdb/data/structures/divided/pdb/*`
  `$SCHRODINGER_THIRDPARTY/database/pdb/data/structures/divided/pdb/*`
  and is in the right format (UNIX compressed, `pdb*.ent.gz`).

D.2.2 Warnings in Homologs Table

These messages appear in the Warning column of the Homologs table for sequences that are likely to be unusable for model building:

Many missing residues

- Too many consecutive missing residues to guarantee model building will succeed.

!UNUSABLE: Alternate residue type

- The residue type was undetermined, causing the file to include the residue twice but with different identity (e.g. ALA20 and VAL20). This is not supported by Prime.

!UNUSABLE: Bad SEQRES

- General problems with the SEQRES.

!UNUSABLE: CA only

- File contains only alpha C coordinates.

!UNUSABLE: Molecule *molname* should be labeled as HETATM

- A non-protein molecule has been listed as ATOM when it should be listed as HETATM. ATOM records are reserved for proteins only.

!UNUSABLE: SEQRES/ATOM mismatch

- The sequence listed in the SEQRES records does not match the actual sequence in the ATOM records.

!UNUSABLE: SEQRES is missing residues

- There are residues in the ATOM records that are not listed in the SEQRES records. PDB files require that the ATOM records be a subset of the SEQRES records.

!UNUSABLE: SEQRES numbering is wrong

- The number of residues in a particular chain does not agree with the number of residues shown in columns 14 to 17 of SEQRES.

D.3 Edit Alignment

Check Alignment

When you click Next at the end of the Edit Alignment step, the Check Alignment warning may appear. The warning lists possible alignment problems that you may want to fix before continuing to the next step. For example, there may be gaps in secondary structure elements of the template, or gaps in the query that are aligned to secondary structure elements of the template.

If you do not want to make corrections to the alignment, click Continue to go to the next step. To make corrections based on the information in the error message, make a note of the gap locations given, click Cancel to close the message box while remaining in Edit Alignment, and change the alignment accordingly.

D.4 Build Structure

Error messages and warnings for this generally appear in the log of a running job. The log can be viewed in the job Monitor panel until the job is completed.

D.5 Refinement

Error messages for Refinement generally appear in the log of a running job. The following is an example of a warning in the log of a loop refinement.

Invalid Features

When a loop refinement begins, a validation program checks the loop features of the structure. Apparent errors are reported in a warning dialog box. You may choose to Run All Features, Run Only Valid Features, or Cancel. It is recommended that you make a note of the invalid features, click Cancel, and correct the structure if appropriate.

Getting Help

Schrödinger software is distributed with documentation in PDF format. If the documentation is not installed in `$SCHRODINGER/docs` on a computer that you have access to, you should install it or ask your system administrator to install it.

For help installing and setting up licenses for Schrödinger software and installing documentation, see the *Installation Guide*. For information on running jobs, see the *Job Control Guide*.

Maestro has automatic, context-sensitive help (Auto-Help and Balloon Help, or tooltips), and an online help system. To get help, follow the steps below.

- Check the Auto-Help text box, which is located at the foot of the main window. If help is available for the task you are performing, it is automatically displayed there. Auto-Help contains a single line of information. For more detailed information, use the online help.
- If you want information about a GUI element, such as a button or option, there may be Balloon Help for the item. Pause the cursor over the element. If the Balloon Help does not appear, check that Show Balloon Help is selected in the Maestro menu of the main window. If there is Balloon Help for the element, it appears within a few seconds.
- For information about a panel or the tab that is displayed in a panel, click the Help button in the panel, or press F1. The help topic is displayed in your browser.
- For other information in the online help, open the default help topic by choosing Online Help from the Help menu on the main menu bar or by pressing CTRL+H. This topic is displayed in your browser. You can navigate to topics in the navigation bar.

The Help menu also provides access to the manuals (including a full text search), the FAQ pages, the New Features pages, and several other topics.

If you do not find the information you need in the Maestro help system, check the following:

- *Maestro User Manual*, for detailed information on using Maestro
- *Maestro Command Reference Manual*, for information on Maestro commands
- *Maestro Overview*, for an overview of the main features of Maestro
- *Maestro Tutorial*, for a tutorial introduction to basic Maestro features
- *Prime Quick Start Guide*, for a tutorial introduction to Prime
- Prime Frequently Asked Questions pages, at https://www.schrodinger.com/Prime_FAQ.html
- Known Issues pages, available on the Support Center.

The manuals are also available in PDF format from the Schrödinger Support Center. Local copies of the FAQs and Known Issues pages can be viewed by opening the file `Suite_2009_Index.html`, which is in the `docs` directory of the software installation, and following the links to the relevant index pages.

Information on available scripts can be found on the Script Center. Information on available software updates can be obtained by choosing Check for Updates from the Maestro menu.

If you have questions that are not answered from any of the above sources, contact Schrödinger using the information below.

E-mail: help@schrodinger.com
USPS: Schrödinger, 101 SW Main Street, Suite 1300, Portland, OR 97204
Phone: (503) 299-1150
Fax: (503) 299-4532
WWW: http://www.schrodinger.com
FTP: ftp://ftp.schrodinger.com

Generally, e-mail correspondence is best because you can send machine output, if necessary. When sending e-mail messages, please include the following information:

- All relevant user input and machine output
- Prime purchaser (company, research institution, or individual)
- Primary Prime user
- Computer platform type
- Operating system with version number
- Prime version number
- Maestro version number
- mmshare version number

On UNIX you can obtain the machine and system information listed above by entering the following command at a shell prompt:

```
$SCHRODINGER/utilities/postmortem
```

This command generates a file named *username*`-`*host*`-schrodinger.tar.gz`, which you should send to help@schrodinger.com. If you have a job that failed, enter the following command:

```
$SCHRODINGER/utilities/postmortem jobid
```

where *jobid* is the job ID of the failed job, which you can find in the Monitor panel. This command archives job information as well as the machine and system information, and includes input and output files (but not structure files). If you have sensitive data in the job launch directory, you should move those files to another location first. The archive is named *jobid*`-archive.tar.gz`, and should be sent to help@schrodinger.com instead.

If Maestro fails, an error report that contains the relevant information is written to the current working directory. The report is named `maestro_error.txt`, and should be sent to help@schrodinger.com. A message giving the location of this file is written to the terminal window.

More information on the `postmortem` command can be found in Appendix A of the *Job Control Guide*.

On Windows, machine and system information is stored on your desktop in the file `schrodinger_machid.txt`. If you have installed software versions for more than one release, there will be multiple copies of this file, named `schrodinger_machid-`*N*`.txt`, where *N* is a number. In this case you should check that you send the correct version of the file (which will usually be the latest version).

If Maestro fails to start, send email to help@schrodinger.com describing the circumstances, and attach the file `maestro_error.txt`. If Maestro fails after startup, attach this file and the file `maestro.EXE.dmp`. These files can be found in the following directory:

```
%USERPROFILE%\Local Settings\Application Data\Schrodinger\appcrash
```

Glossary

alignment—The optimal matching of residue positions between sequences, typically a query sequence and one or more template sequences.

anchor—A constraint on alignment set at a given residue position. Alignment changes must preserve the query-template pairing at that residue until the anchor is removed.

ASL—Atom Specification Language.

cofactor—The complement of an enzyme reaction, usually a metal ion or metal complex.

Comparative Modeling—Protein structure modeling based on a query-template match with a substantial percentage of identical residues (usually 50% or greater sequence identity).

composite template—A type of template used in the Threading path, produced from the core (invariable) and variable regions of a family of structurally similar proteins.

constraints—Tools to keep regions of a sequence (alignment constraints) or structure (during minimization) in a particular configuration.

deletions—The residues missing from a query sequence that are present in a template sequence.

Fold Recognition—The use of secondary structure matching and profiles generated from multiple sequence/structure alignments to find templates when sequence methods are unsuccessful.

gaps—The spaces in an alignment resulting from insertions and deletions.

HETATOMs—The atoms of residues, including amino acids, that are not one of the standard 20 amino acids. In PDB files, HETATM.

homolog—A sequence/structure related to the query sequence; i.e., a sequence with many of the same residues in the same patterns as the query sequence. Usually these sequences are derived from the same family and may have similar function.

insertions—The extra residues found in a query sequence that are not found in a template sequence.

ligand—A small molecule that binds to a protein; e.g., an antigen, hormone, neurotransmitter, substrate, inhibitor, and so on.

loop—A region of undefined secondary structure.

project—A collection of related data, such as structures with their associated properties. In Prime a project comprises one or more *runs* (executions of the Prime workflow). The project may include data that does not appear in the *project table*.

Project Table—The Maestro panel associated with a project, featuring a table with rows of entries and columns of properties. In Prime, project table entries are usually model structures.

query sequence—A sequence of unknown structure or fold.

Ranking Score—The score used to rank composite templates derived from different seed templates. Generated by the Global Scoring Function in the Threading Path.

refinement—An improvement of a model structure through energy-based optimization of selected regions.

run—A single execution of the Prime workflow using a particular set of choices (of templates, of paths, and of settings). Each run belongs to a *project*. Runs cannot be saved without saving the project to which they belong.

SSA—Secondary structure assignment.

SSP—Secondary structure prediction.

template sequence—A sequence of known structure and fold used as a basis for building a model of the query.

Threading—A structure prediction process in which Fold Recognition is used to define templates, then backbone models are built via alignment to composite templates and refined. May be used when query-template sequence identity is low.

Workspace—The open area in the center of the Maestro window in which structures are displayed.

Z-Score—Measures the compatibility of the query sequence with the model structure, relative to the compatibility of randomly shuffled sequences of the same composition.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

120 West 45th Street, 29th Floor
New York, NY 10036

101 SW Main Street, Suite 1300
Portland, OR 97204

8910 University Center Lane, Suite 270
San Diego, CA 92122

Zeppelinstraße 13
81669 München, Germany

Dynamostraße 13
68165 Mannheim, Germany

Quatro House, Frimley Road
Camberley GU16 7ER, United Kingdom

SCHRÖDINGER®